

Xpos're 2.0

Users Guide
April 2012

Introduction

Xpos're: authoring enhanced publications

An [enhanced publication](#) (EP, or rich Internet publication, or shortly a RIP) is a regular scholarly publication with additional material, like data, models, algorithms, illustrative images, metadata sets or post-publication information such as comments or rankings. There is a growing interest in publishing results together with the underlying data and interactive facilities. This enables the reader to explore research objects, to experiment with the included data sets and to validate the author's conclusions.

The Xpos're tools are designed for authoring and displaying EPs. The software comprises a Flash-based document reader for text encoded in XML, and a set of extensions (plug-ins) that extend the basic functionality of the reader. The extensions are used to display specific types of multimedia with additional functionality, such as zoomable images, videos and interactive maps.

The document reader can also generate output in HTML, in two flavors, namely a slide based version that uses Javascript, and a plain HTML text (single page), which can be used to create e-books with the help of programs such as [Calibre](#). Xpos're is very flexible and not bound to the common publication structure of specific disciplines.

Interactivity means reading on-line, which requires a suitable text: not too long, easily scanned by the human eye and preferably with different levels of detail. Essentials are communicated first, and detailed information should be available for those who want to learn more. Xpos're encourages but does not enforce such a writing style. Each slide may start with a short summary, followed by the core message and optional continuation text of any length. An other feature of EPs is integration: related parts of the text are linked to each other, discussions on findings are connected to the related data, and vice versa. Xpos're supports these requirements by various types of internal links.

An Xpos're publication can be used as extra "showcase" on a personal or institutional website in combination with a PDF file of an article, which has been published elsewhere. Each slide can be linked to the related page in the PDF, while the added data and functionalities provide a deeper insight in and greater visibility of the research project.

Conventions in this documentation

In program code and in XML text underlined items in black are placeholders, to be replaced with appropriate terms.

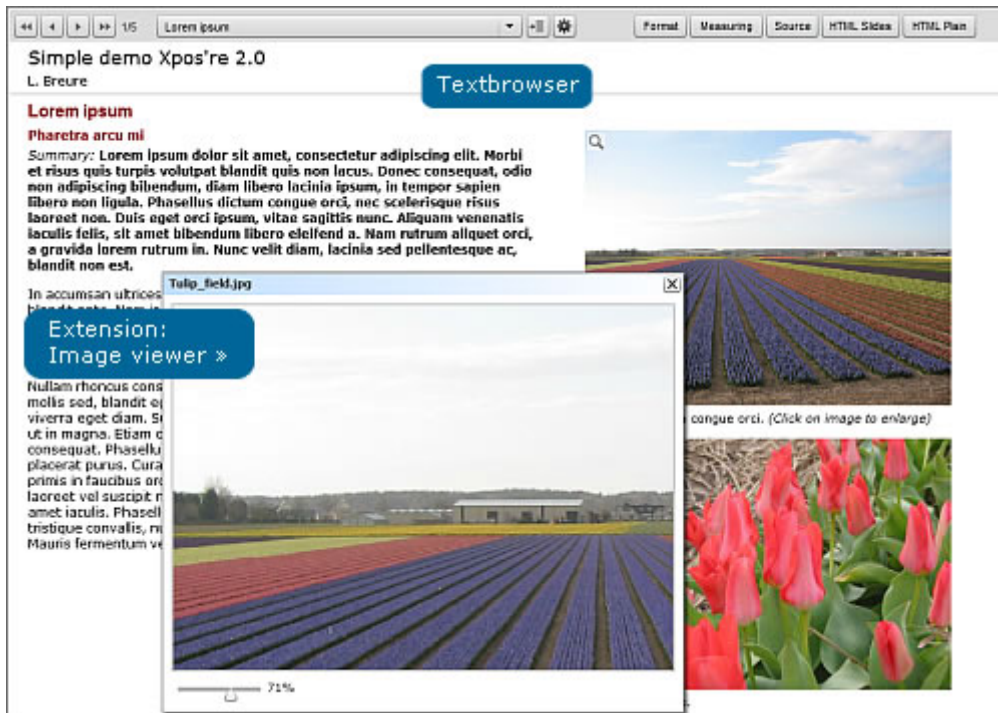
Features

1. **Sustainability:** the publication text is highly sustainable. Its structure and interactivity is defined in XML, a widely accepted standard for encoding content. The input of the Xpos're extensions is in XHTML.
2. **Cross-platform:** the XML-text is primarily rendered in Flash, which ensures a high graphic quality and reliable interactivity. HTML is supported for non-Flash platforms and for users who do not want to use Flash.
3. **Flexibility:** an author can define the complete layout (like size and position of the text and the visual elements) through XML tags.
4. **Referencing:** normal hyperlinks can be inserted into the text and parts of the publication can be connected by cross references (end notes, table of contents, links to other text sections).
5. **Extensible architecture:** The functionality of the document reader itself is simple. Essentially, the application is a special browser, which allows the user to go through the set of slides that make up the enhanced publication, and to jump from one place to another using hyperlinks. However, this basic functionality can be extended by means of additional widgets (extensions), for example an image gallery, interactive map or a viewer for a database table. Most of the extensions that come with Xpos're, are also small Flash programs. However, HTML applications can be used as well and displayed as an overlay on top of the reader. Extensions can be downloaded and included by simply inserting a hyperlink or custom button. There is no installation procedure.
6. **Client-side:** All operations are on the local computer; there is no content management system on the server or other back-end system. This gives the author complete control over his/her enhanced publication. (S)he can create the content, download extensions as required, upload everything, which completes the authoring process.
7. **Easy learning curve:** users with basic knowledge of HTML and XML can create their own enhanced publication using an XML editor and basic image processing software. The download package contains a simple demo, which is a good starting point. The Xpos're website contains more advanced examples, which can be inspected to learn more.
8. **Semantics:** To retrieve publications from the web, semantics are to be accessible to search engines. This is even more true for enhanced publications, which carry data sets, additional illustrations and interactive modules. To enhance findability RDF-metadata (in the format of [OAI-ORE](#)) are added to the enhanced publication. These metadata can be generated on basis of the XML-text prepared for the document reader.
9. **Interoperability** with other software: The document reader will follow current trends in software development for enhanced publications and is designed to work together with other packages for data display.
10. **Multi-discipline:** The document reader does not make any assumptions on the publication structure. The system is particularly developed with the publication genres of the humanities in mind. However it can support other disciplines as well.

Document reader

Textbrowser

The main program is the document reader (*Textbrowser*). It displays an enhanced publication on the user's computer and it supports the authoring process. It acts as a master, which calls certain extension programs as slaves. These extensions provide additional functionality such as the display of video's, image galleries and interactive spreadsheets.



The textbrowser, showing a slide with the extension Image Viewer activated.

All data are kept *outside* the Flash programs (this in contrast with the common association with a Flash application as a movie in which everything is embedded). Data are first downloaded and processed before the publication is displayed, which guarantees good performance. This pertains in particular to images, which require much more download time than text. This so called *preloading* may take several seconds (you will see a progress bar, indicating the percentage of downloaded data). Next time loading is much faster, because all these data are stored in the cache of the browser on your computer.

Like all Flash applications embedded in web pages, the document reader requires a browser plug-in. Nowadays this so-called Flash reader is already installed on the majority of computers; otherwise it can be downloaded for free from [Adobe](http://www.adobe.com).

Slide as basic unit

A slide is the basic unit of display in Xpos're. It has three areas. There are two text blocks and a list of visuals:

- **Text1** is supposed to contain the main text.
- **Text2** subsidiary text.
Alternatively, both text blocks can be used to create two text columns. If text length exceeds the size of the text block defined, two scroll buttons will appear. However, it is recommended, for aesthetic reasons, to avoid this situation and to use continuation text instead of scrolling. Each slide can have one **continuation text**, which is accessed through an [internal hyperlink](#).
- By default **visuals** are displayed as a list, which can be either horizontal or vertical, depending on the template used. However, the author is free to position in individual image elsewhere on the slide by specifying its coordinates.

Normally, visuals are stored in a subdirectory of the folder where the publication itself resides, but also images on the Web can be included. This applies also to images used in native Xpos're extensions, such as the image gallery and the image viewer. Each visual may have a caption (text explaining the picture). The caption is technically part of the visual.

In case of a vertical list, all visuals are resized to the specified width. It is left to the author to determine the right number of visuals per slide. Images do not need to be of the same size or aspect ratio. Although the program can resize images rather well, a big difference between the original size and the display size of an image is to be avoided: large images take more time to download and resizing is less precise than resampling (the resized image may not look nice). So, the width and height of the images should come close to their display size.

For technical details, see topic [Adding a slide](#)

Input and output

Input

Textbrowser takes two XML files as input:

1. The publication text, in case of the demo: `simple_demo.xml`. This must be validated using `Publication.dtd`.
2. The file `Textbrowser.xml` which contains instructions how Textbrowser should handle the text and where to find things. For example, it contains the path to the folder where the images are stored (e.g. `visuals/`). This must be validated using `Textbrowser.dtd`.

In addition the images referenced in the publication text are also part of the input.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE doc SYSTEM "Publication.dtd">
3 <doc>
4   <!-- Size of slide: 950 x 700 -->
5   <meta>
6     <mainTitle>Simple demo Xpos're 2.0</mainTitle>
7     <author>L. Breure</author>
8     <creator>L. Breure</creator>
9     <modified>2 March 2012</modified>
10    <description>
11      <p>This demo shows the basic features of XPOS'RE 2.0</p>
12    </description>
13  </meta>
14
15  <slides>
16    <!-- ----- -->
17    <slide id="intro">
18      <!-- Note that 'wright' is the default template, see Textbrowser.xml for definition.
19       In this template visuals have a default width of 350px. That is what we are using in this slide.
20      -->
21      <pdfPage>1</pdfPage>
22      <title>Lorem ipsum</title>
23
24      <visual>
25        <url>Tulip_field.jpg</url>
26        <onClick href="event:add#ImageViewer.swf#Tulip_field.jpg"/>
27        <cap>Phasellus dictum congue orci. <i>(Click on image to enlarge)</i></cap>
28      </visual>
29
30      <visual>
31        <url>Red_tulips.jpg</url>
32        <cap>Sodales nisl quis.</cap>
33      </visual>
34
35      <text1>
36        <h2>Pharetra arcu mi</h2>
37        <p>
38          <i>Summary:</i>
39          <b>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi et risus quis
40            turpis volutpat blandit quis non lacus. Donec consequat, odio non adipiscing
41            bibendum, diam libero lacinia ipsum, in tempor sapien libero non ligula.
42            Phasellus dictum congue orci, nec scelerisque risus laoreet non. Duis eget
43            orci ipsum, vitae sagittis nunc. Aliquam venenatis iaculis felis, sit amet
44            bibendum libero eleifend a. Nam rutrum aliquet orci, a gravida lorem rutrum
45            in. Nunc velit diam, lacinia sed pellentesque ac, blandit non est. </b>
46        </p>
47      </text1>
48    </slide>
49  </slides>
50 </doc>
```

Example of XML-input (Simple_demo.xml)

Structure of the text

The text of the enhanced publication is encoded in XML and is divided into *slides*. A slide is comparable with a slide in PowerPoint except that it contains full text in stead of key words and short phrases.

Each slide contains the text of a single topic, equivalent to a (larger) subsection in a printed paper, plus some metadata. It comprises the following elements:

- The title of the slide.
- The visuals to be displayed next to the text block. Visuals may be real images of type JPEG, PNG or GIF, or a Flash-file (SWF), which is treated as a visual as well. In some cases, a SWF can be a handy format for a graphic illustration, provided the author can dispose of a Flash authoring program.

Note: the SWF must be an ActionScript3 file (ActionScript2 files are not correctly loaded). Do **not** use SWF-visuals if you want to generate HTML-versions; SWF files are not displayed in the generated HTML text.

- Optional format instructions: where is the text to be displayed on the screen, to what size are the visuals to be scaled etc. (if the author wants to deviate from the template).
- The number of the related PDF-page, if the enhanced publication is accompanied with the full text of the printed publication in a PDF-file.

Layout

The content's appearance is defined as follows:

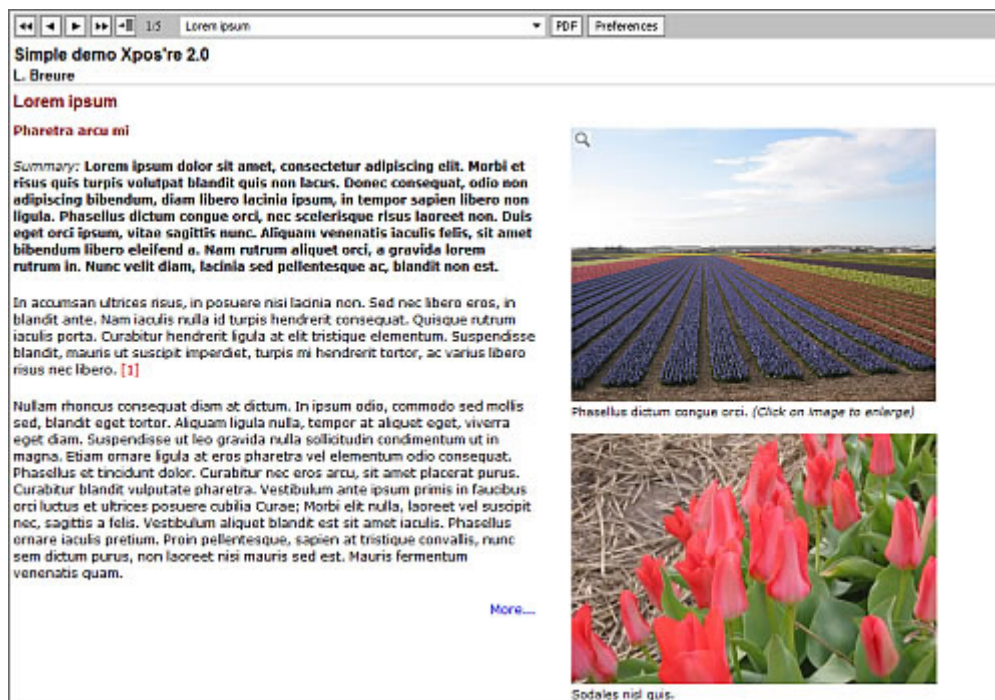
1. The style (font family, size, hyperlink style etc.) through a related style sheet (CSS), for the demo: **RIP-Flash.css**.
2. The layout (where goes what on the screen) in defined in two ways:
 - Through templates, specified in **Textbrowser.xml**. Currently, three templates are predefined (but you are free to define and add your own):
 - **vleft**: visuals left, and text right
 - **vright**: visuals right, text left. Note, that **vright** is the default template.
 - **cols**: no visuals, but two text columns
 - Within the publication text, through attributes, such as **x**, **y**, **width**.

The templates define the default values for several attributes. The template may vary from slide to slide: the desired template is specified through an attribute of the `<slide>` tag. Irrespective of the template chosen an author may add attributes to text and visuals or change attribute values where appropriate, which makes, that these elements can be placed anywhere on the slide as required.

Cross platform

There are several reasons why a HTML version is wanted: one may dislike the textbrowser's user interface, hardware may not support Flash, or a more accessible / printable document is required. Xpos're offers two ways to convert the EP to HTML:

1. [HTML slides](#): this generates a look-alike of the Flash version, with almost the same functionality. It is based on CSS and Javascript.
2. [Plain HTML](#): this produces a single HTML document with a minimum of formatting and only based on CSS (no Javascript). This code can be used as input, for example, for a program as [Calibre](#) to create an e-book file.



The HTML-version of the slides generated by the textbrowser.

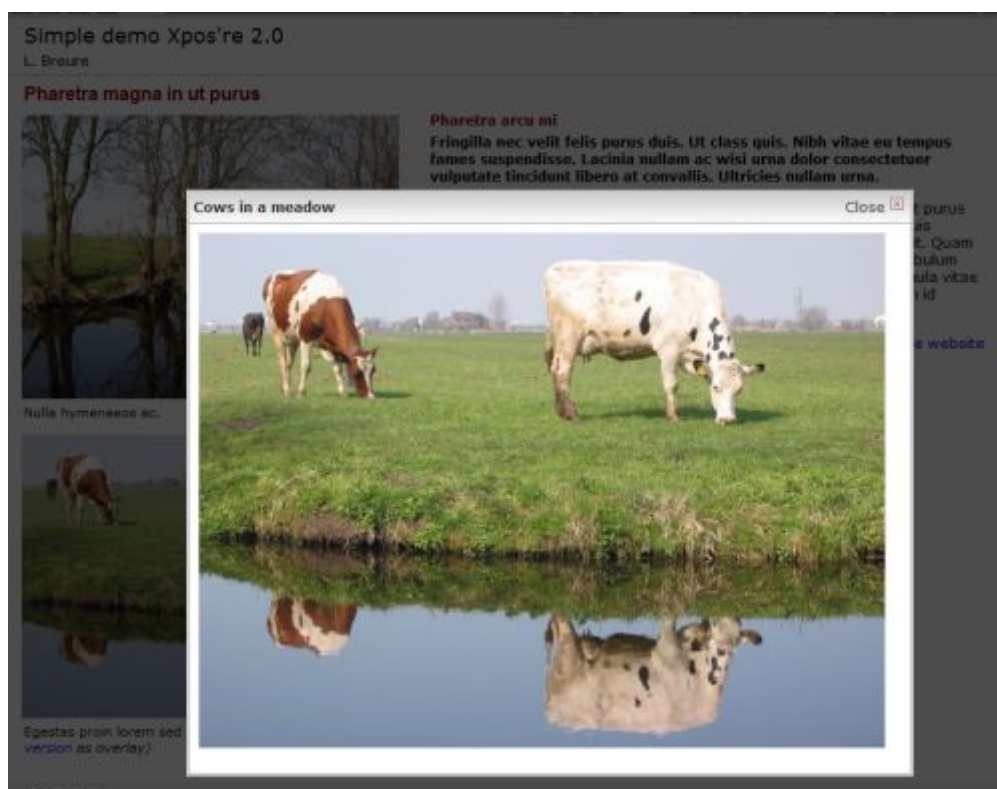
Note: Both of them, the HTML slides and the plan HTML version, use files included in the subdirectory **RIP-HTML**: do not forget to copy this folder when you install the package.

Extending the functionality

The basic functionality of the document reader consists of browsing through the set of slides that make up the enhanced publication, and jumping from one place to another, using hyperlinks. However, this functionality can be extended in two ways:

1. The textbrowser can open native Xpos're **extensions** (to be downloaded from the Xpos're website) such as an image gallery, interactive map or a table viewer. Using extensions is simple: the author inserts a special link (so called *event-link*) into the text; when the reader clicks on the link, the extension is displayed. Most of these extensions are also Flash programs and are displayed as pop-ups within the textbrowser.
2. There many web-based applications for displaying numeric data, maps, time lines, image galleries etc. Just like in an HTML-text the author can insert hyperlinks in the XML code to link to any other web document and to display it in a new window or tab. A minor drawback of this method is the separation of illustrative data from the main text. Up to a certain extent this can be redressed by using **overlays**. Xpos're uses [GreyBox](#) for this purpose and provides a special event-link to overlay web pages and images on top of the textbrowser.

Note: The files in subdirectory `greybox` are required for this purpose. Do not forget to copy this folder when you install.



Displaying a picture as overlay on top of the textbrowser.
Note that the transparency is not rendered correctly by all browsers.

How to use Xpos're

Currently most scholarly journals are still rather conservative in set-up. Publishing an on line version of the publication in PDF format is quite common (even for printed journals). In addition, some publishers have a separate multimedia section on their website. As a rule all publications have to comply with the journal's template, so, for extras authors are completely dependent on the facilities offered by the journal of their choice.

Xpos're may be an attractive option for an additional publication of an article on the author's own website or on that of the faculty. This makes it possible to include much more material than a regular journal publication allows and it offers a chance to reach a broader audience.

One may do one of the following:

1. Publish a shorter version of the article, and link each slide to the related page in the PDF of the original publication. In this option the emphasis is on the extras, such as images, videos, data.
2. Re-publish the entire original text (assumed that it is allowed by the copyright rules of the journal in which it has appeared) enhanced with additional material. Also in this case synchronization of slides and PDF pages can be useful.

In the current situation credits are primarily earned by publishing in highly ranked journals. However *exposure* of your work may be greatly enhanced through a "show case" as Xpos're can produce.

Sustainability

The volatile Web

One of the problems of the Web is its volatile character, which is particularly a problem with enhanced publishing. How can we make sure that the entire enhanced publication is still legible and completely functioning over ten, twenty or more years? PDF is so widely used that we don't need to have serious concerns about this format.

Meta data and persistent identifiers

Sustainability of web publications has different aspects. One of them is the use of persistent identifiers and adequate metadata to ensure the coherence and findability of text, data and multimedia. Xpos're supports the generation of a so-called *resource map*, encoded in RDF (OAI-ORE, see [Open Archives Initiative](#)), which meets these semantic requirements. This resource map can be automatically generated by means of an XSLT file, which is to be downloaded from the Xpos're website.

However, the metadata are only a part of the problem. The display of additional material requires dedicated software, which is not guaranteed for the (distant) future or does not run on certain platforms -- most well-known is the fact that Flash is not supported on an iPad. The creation of interactive spreadsheets may require, that data are uploaded to a third party website. Are we sure that the service of rendering these data will not be discontinued at a certain moment?

The principle of 'gentle fall-back'

We can not solve all these insecurities beforehand, however some precautionary measures can and should be taken. Xpos're itself is based on the principle of *gentle fall-back*: all input data are in a basic format that can be displayed by a web browser without special software. If a more advanced software solution does not work, it can be switched off, which means that the reader falls back on a more basic rendering. That's also why the document reader can generate two different flavors of HTML, one more advanced and one very basic, even without Javascript.

Other examples:

- The use of overlays is optional, not only for the author, but also for the reader. The user can switch off this facility in *Preferences*; in that case the overlaid document is simply displayed in a separate window.
- The HTML slides allow the use of Flash extensions such as the image viewer. When Flash is not supported, a reader can deactivate the use of Flash extensions and the image will simply appear in a new tab.
- Moreover, all input of the native Xpos're extensions is basic HTML, so this enhancement can be displayed even without the textbrowser.

What authors should do

So Xpos're has taken its share in the precautions. However, the author has his/her own responsibility. There is a wealth of interesting software for displaying added material on the Web: generators for image galleries, slide shows, time lines, interactive maps, interactive spreadsheets and graphs. Xpos're slides can get linked to such web documents and display them, for example, in overlay mode, but to keep it safe, the author has to add also extra links to the data in a sustainable format. For example, in this approach an interactive spreadsheet displayed through [EditGrid](#) needs an accompanying link to the same data in tab or comma separated format, preferably stored on a server controlled by the author or by the organization with which (s)he is affiliated.

Getting started

Downloading

The software can be downloaded from the Xpos're website: <http://xposre.nl> (note, that the apostrophe is missing in the URL!). The basic package contains a simple demo (`simple_demo.xml`) with pseudo-text, starting with "Lorem ipsum". The 'lorem ipsum' text is typically a section of a Latin text by Cicero with words altered, added and removed that make it nonsensical in meaning and not proper Latin; it is not intended to have any meaning. The whole demo consists of this kind of filler text, which is commonly used to demonstrate the graphic elements of a document and the visual presentation, such as font, typography, and layout.

The download package contains only one native Xpos're extension: the image viewer. More extensions can be downloaded from the Xpos're website when required.

Installing Xpos're

Installation is very simple: unzip the download file and copy its content to the hard disk of your computer. Keep the directory structure as present in the zip file.

Make sure that your computer has a recent Flash player installed:

- for a check, go to http://www.adobe.com/software/flash/about/and_downloading,
- or for downloading, go to: <http://get.adobe.com/flashplayer/>

Getting started

You need a text-oriented XML editor to prepare input data and to view the various XML files. There are different sorts of XML editors: some display the XML text only as a tree, which is not convenient for textual publications. An other important feature is validation against a DTD, which is not supported by every XML editor. The Xpos're website contains more information about useful software. Most commercial editors come with a trial version and offer low prices academic license. An acceptable free XML editor is [Jedit](#), provided you download and install the XML extensions for this program.

The following steps will help you to get started:

Explore the demo:

1. Open the file `Textbrowser.html` in your browser.
2. Browse through the slides using the arrow keys on screen or the arrow keys on your keyboard.
3. Click on links in the text and the clickable images, indicated by the magnifier (note that the image, not the magnifier itself is clickable).

IMPORTANT: For safety reasons the Flash player will not open normal hyperlinks to other web documents when you are working off-line as in this exploration. Nothing will happen when you click on a link. You must first tell the Flash player that the folder that contains the Flash file, is "trusted". Do this:

1. Open `Textbrowser.html` in your browser.
2. Right click somewhere on the Flash application: the context menu appears.
3. Select 'Global Settings...'
4. Click on tab 'Advanced'
5. Scroll down to the button 'Trusted Location Settings'
6. Add the folder that you want to be considered as trusted (all subfolders are automatically trusted as well)

Experiment: modify the demo:

1. Make a backup of the file `simple_demo.xml`.
2. Open this file in your XML browser.
3. Modify existing text in the `<text1>` tag of a slide.
4. Validate the modified XML text. If the XML text is not valid, you will get later an error message in the `textbrowser`.

5. Reload (refresh) `Textbrowser.html` in your browser and view the results.
6. Replace an existing image. Store the new image in the folder visuals. If the image is not found, an error message is displayed.
Note that the new image does need to be of exactly the same size. Xpos're will resize image. However, you will notice that in some cases prior resizing / resampling the image will give better results.
7. Add a new slide to the text in your XML editor. You may copy an existing slide, but make sure that you change the id-attribute of the slide. All id-values must be unique. See also topic [Adding a slide](#).
8. Go back to step 4.


Uploading the enhanced publication to a web server is simple as well: copy all files to the public HTML folder of the server.

User interface


The button bar

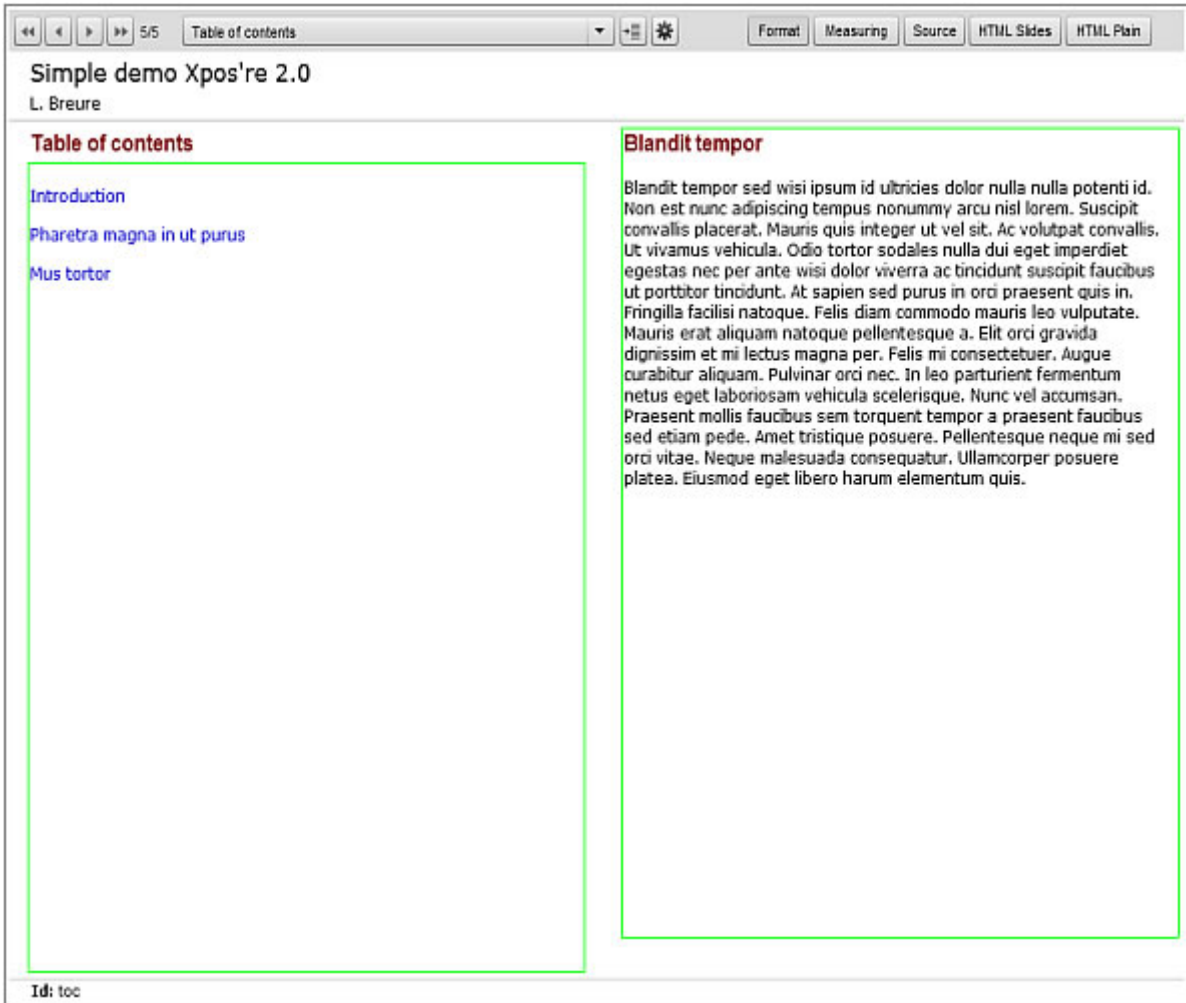


Button bar of the Flash version of the textbrowser.

- **Browsing:** to move from one slide to an other use the arrow buttons on the navigation bar or the related keys of the keyboard. Do not forget to select the document reader first (click on the reader area) before pressing the button, otherwise it will not work:
 - Arrow Left or Page Up: previous slide
 - Arrow Right or Page Down: next slide
 - Double Arrow Left or up-arrow or Home: first slide
 - Double Arrow Right or down-arrow or End: last slide
- **Jumping:**
 - Use the drop down box in the button bar, or
 - the table of contents  to move quickly from one slide to an other. The table of contents is a regular slide (with the fixed `id="toc"`), which must be explicitly provided by the author. If it is missing, the Table of Contents button is disabled.
 - Navigation links: the author may have inserted hyperlinks in the text to other slides. These links are of the same type as the links in the table of contents.

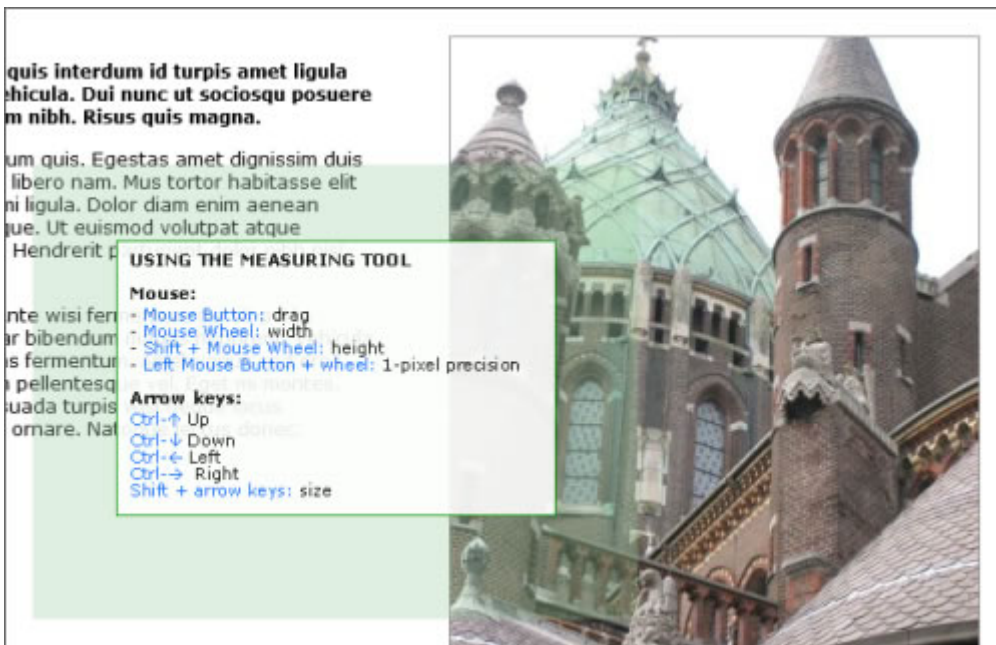
Back-button of the browser: be aware, that you can not use the browser's previous and next button to navigate through the slides. The browser buttons are used to navigate through web pages. The document reader is an application embedded in a single web page and, therefore, requires its own navigation buttons.

- **Preferences:**  This button opens the Preference window:
 - 'Switch between authoring and reading mode': In reading mode the buttons after 'PDF' are hidden.
 - 'Use overlays for external links': if checked, GreyBox overlays are used, otherwise the overlaid document is displayed in a new window.
 - 'Show help with measuring tool': if checked a text panel with instructions how to move and to resize the measuring tool is displayed on activation. It always disappears as soon as the measuring tool is moved (see below).
- **PDF:** This buttons opens the associated PDF file provided that
 1. this file is correctly specified in `Textbrowser.xml` (otherwise this button is hidden) and
 2. the related PDF page is specified in the slide in the publication XML file (e.g. in `simple_demo.xml`)
- **Format:** This displays an outline of the layout elements of a slide:



Showing the format of a slide: size and position of layout elements.

- Measuring:** This displays the measuring tool, i.e. a transparent colored rectangle, which can be moved around over the slide and resized in order to measure existing objects such as text blocks and images or to find a new position and space for these elements. Unless turned off in *Preferences*, it appears with a text on top explaining the keys to be used with this tool. Position and size are displayed at the bottom of the screen in a format that can be directly copied and pasted into the XML code.



The measuring tool activated.

- **Source:** The XML source text is opened in an other browser window.
- **HTML Slides:** This opens a new window with the code of the [HTML slides](#) version.
- **HTML Plain:** This opens a new window with the code of the [plain HTML](#) version.

Other features

Large images

Some images displayed are clickable; they have a magnifier in the left upper corner. On click the image viewer extension is displayed, which has a zoom option.

Windows

The document reader has its own draggable pop-up windows (for example for notes and literature references), which can be closed by clicking on the x-button. All open windows are closed, when the user navigates to an other slide.

Printing

Use the browser's print button to print *or* the plain HTML-version, which is most suitable for printing the text as a whole.

Size of the application and screen size

The application has a fixed size (950 x 720 pixels), which fits well on an average computer screen and beamer display. Nevertheless in some cases the Flash area may not be completely visible, which enforces scrolling. To make the Flash area completely visible, you may do one of the following:

1. Hide some browser toolbars to enlarge the display space.
2. Switch to the browser's full screen mode (recommended).
3. Zoom out in the browser.

Authoring

Prerequisite knowledge and skills

- Basics of XML en HTML:
 - Understanding XML fundamentals
 - Being able to use an XML-editor
 - Understanding a DTD and validating an XML-text
 - Formatting in HTML (<p>, , , <style>, etc.)
 - Understanding cascading style sheets
- Simple image processing:
 - Capturing screen shots
 - Enhancing image quality (brightness, sharpness etc.)
 - Cropping images
 - Resizing and resampling images
 - Adding text and symbols (shapes) to images

Software

For authoring content you need some additional software:

- A text-oriented XML-editor that can validate the text against a DTD.
- Image processing software to create and to modify images.

Planning

Authoring requires not only some tools, but also a sound planning. The best starting point is an already published text. Select a publication, together with multimedia material and data. Take the following steps as preparation for authoring the enhanced publication:

1. If it has not yet a suitable subdivision divide the publication into sections. An entire section or a summary of a section becomes the text of a slide. If it does not fit on a slide, you may use continuation text (<more> tag).
2. Gather illustrations in high quality, preferably one or more for each slide. In most cases, two illustration will fit on a slide, however, you can display more by means of an image gallery or by links to external images.
3. Consider data to be added to the publication.
4. Although any number of slides can be added, it is a good idea to make the publication not too long. Working with a very long XML text may be inconvenient. [Preloading](#) a large number of images will take time and consume memory space. Instead, consider more publication files when the text is, for example, as large as a book and bring them together in an HTML file, using a frame set. Then each "chapter" may be opened when required.

Defining the publication

1. The simple demo may be an easy starting point: modify and extend the existing text (see [Getting started](#)).
2. Specify the new enhanced publication (its metadata) in `Textbrowser.xml` and validate the text against `Textbrowser.dtd`:

```
== CURRENT PUBLICATION ==
-->
<mode>authoring</mode> <!-- authoring | reading | protected -->
<urlText>Simple_demo.xml</urlText> <!-- the XML version of the publication -->
<urlPDF></urlPDF> <!-- the PDF version of the publication -->
<urlCSS>RIP-Flash.css</urlCSS> <!-- cascading stylesheet -->
<imagePath>visuals/</imagePath> <!-- relative file path to the images -->
<welcomeText> <!-- start-up text, displayed during preloading -->
  <p><b>ENHANCED PUBLICATION</b><br/>XPOS'RE 2.0<br/><br/>
<font size="-3">Leen Breure 2012</font></p></welcomeText>
<scrollDelta>1</scrollDelta> <!-- number of lines scrolled in a single step by scroll buttons; default: 10 -->
<measuringHelp>on</measuringHelp> <!-- keyboard shortcuts for measuring tool; on | off -->
<useOverlay>on</useOverlay> <!-- on | off: uses Greybox if href="event:overlay#title#url" -->
<!--
```

The section in `Textbrowser.xml` where the enhanced publication is specified.

3. You must work in 'authoring mode', in which all buttons on the button bar are visible.
 - In 'reading mode' only buttons required for reading are displayed.
 - 'Protected mode' is similar to 'reading mode' except for the fact that the user can not switch to authoring mode.

Adding a slide

1. Open your publication text, or, if you start from the demo, `simple_demo.xml`.
2. **Slide:** A slide is defined in the tag `<slide>` (all slides are contained in the tag `<slides>`; do not confuse both). The tags embedded in `<slide>` are optional, except `<title>`. Most tags are to be added in a fixed order; see [DTD](#) for details. Most XML editors are helpful by displaying a list of allowed tags in a particular editing context.

The `<slide>` tag has two attributes:

- **id:** the identifier, whose value must be unique for the current text.
- **template:** optional, if omitted the default template [vright](#) is used.

The value of `id` must start with a letter or underscore, and may contain letters, digits, dots, hyphens and underscores.

3. **PDF-page:** the optional tag `<pdfPage>` contains the number of the related page in the PDF file (that contains the printed version of the publication). Note:
 - You must have defined the url of the PDF in `Textbrowser.xml`
 - A click on the [PDF button](#) will open the PDF file displaying the page specified, at least in most browsers (this has nothing to do with Xpos're, but depends on the PDF browser plug-in).
4. **Title:** the tag `<title>` takes the title of the slide (required).
5. **Visuals:** the `<visual>` tag has optional attributes to define size and position of the image: x, y, width, height, border. If omitted, the default values defined in the [template](#) are used. Using the optional attributes allows to deviate from the default size and position. In templates with a vertical image list, the image size depends on the width of the list. By adding a different x and y value an image can be positioned on any place on the slide. To change the display size, specify either width or height. The image is then resized while the aspect ratio is preserved.
 - `<url>`: the URL of the image (required). The document reader will automatically add the image path as defined in the project through tag `<visuals>`. So, here the simple file name is sufficient. Also images on the Web can be used by specifying the **full** url (starting with `http://...`)
 - `<onClick>` (optional): empty tag, with attribute `href`. This takes the URL, either an external link (`http://...`) or an [internal link](#) (`event:...`). When this tag is present, a visual becomes clickable and a [magnifier](#) symbol is displayed in the upper left corner. Clicking on an image is supposed to display a larger version of the current image, either from the web or through the image viewer widget.
 - `<cap>` (optional, but strongly recommended): the caption of the visual.
 - `<creator>` (optional), to be used in the [resource map](#).
6. **Text:** just like visuals texts may have attributes to define a deviating size and position: x, y, width, height. If omitted, the default values as defined in the template are used. See for border color and background color the topic [Specials](#).
 - `<text1>`: the main text block (optional) or column 1.
 - `<text2>`: additional text (optional) or column 2.
 - `<more>`: use this tag for text continuation. The tag requires a valid XML id. You can display the text contained in `<more>` by inserting an internal link, for example, in `text1`, with a `href` that refers to the `<more>` element.

Note:

- Avoiding scrolling looks better, so the slide text should ideally fit in the `text1` and `text2`. If the text is longer, scroll buttons will appear, however, it is recommended to use continuation text in such a case.

- Inside the text blocks regular HTML tags can be used for basic formatting, like `<p>`, ``, `<h1>`, etc. For details see `Publication.dtd`.
- The style of the text (font family, font size, color etc.) is defined in the cascading style sheet the filename of which is specified in `Textbrowser.xml`.
- In addition to visuals you can also **embed** an image in the text, just like in HTML. However, the text alignment is not so nice in Flash and these images are not preloaded. Only embedding images at the end of the text works well. Store these images also in the visuals directory, but **add the relative image path** (`visuals/myImage.jpg`) in the `scr`-attribute.

It is recommended to validate the text frequently during editing using `Publication.dtd`.

Links and buttons

External links

1. These are regular hyperlinks to a web page ("http://...") or to the HTML document in your project (supply the relative path).
2. These links can be added by ``.
3. You can specify existing frame or iframe as target.

Internal links (event-links)

Internal links use the so-called event-protocol. The syntax is:

``, where:

- o verb is an action to be performed when the user clicks on the link.
- o object is the object on which the action is performed (the parameter may be composite, see table below)
- o parameter influences the way in which the action is executed (the parameter may be composite)

An example of an internal link is:

```
<a href="event:add#ImageViewer.swf#Tulip_field.jpg">Tulips</a>
```

which adds an image viewer (an extension) to the textbrowser, and displays the image file. Note that the requested image must be in the visuals folder.

List of verbs to be used with the event-protocol:

| Verb | Object | Parameter | Meaning |
|---------|---------------------|-------------------------------------|--|
| goto | slide-id | none | Jumps to an other slide: <code>Questions</code> |
| ref | ref-id | none | Opens the references window, displaying the text of the <code><ref></code> having the id ('book1') specified. Here an example of an (end) note: <code>[1]</code> The demo style sheet defines a CSS-attribute .class ref to render the end note reference [1] in red. |
| note | note-id | none | Opens the note window, displaying the end note text: <code>[1]</code> |
| more | none | none | Opens the continuation text of the current slide in a new window. Note that each slide can have only one continuation text. |
| add | url of an extension | depends on extension, mostly an url | Displays the extension as a pop-up: <code>Tulips</code> |
| overlay | <u>title#url</u> | <u>#width#height</u> | Displays document with url as overlay on top of the textbrowser using title in a window of width x height : |

| Verb | Object | Parameter | Meaning |
|--------------|------------------|-----------|--|
| | | | <code>larger version</code> |
| imageOverlay | <u>title#url</u> | none | An image version of previous command: <code>photo</code> |

IMPORTANT: The overlay commands require a relative path to the visuals. The location of the visuals is not automatically added!

Buttons

1. A button may be used instead of a link. In the document reader a button has the same function as a link (internal and external) and takes the same value for the `href` attribute. The advantage of a button is, that it is more prominent visible.
2. A slide may have one or more custom buttons. A button definition applies only to the current slide. You can not define default buttons appearing on all slides or a group of slides. The `<button/>` tag has the following attributes, all required:
 - o label (make sure that it fits the width)
 - o width (in pixels); a button has a fixed height of 22 pixels.
 - o x: x-position
 - o y: y-position
 - o href: this attribute can have the same values as the href in a regular hyperlink (`<a>`).

Special text

Colored text blocks

Text blocks can be transformed into sidebars or message block by using the attributes for border color and background color.

The textbrowser can handle two sorts of color definitions:

1. [Named web colors](#) (for example Gold, CornSilk, Salmon)
2. [Hexadecimal color](#) values (for example #FF0000 for red, and #C0C0C0 for gray).

```
<text2 x="550" y="540" width="385" height="70" borderColor="DarkKhaki"
backgroundcolor="LightYellow">
```

Text on top of an image

Note that text2 has a higher z-index than text1. So, in case of an overlap, text2 is on top of text1 and, as a consequence, text2 may shield any underlying hyperlink in text1 (i.e. the link is not clickable). The visuals have a lower z-index and are below both text blocks (see also FAQ: [background image](#)).

Table of contents

1. The use of a table of contents is optional, but recommended.
2. A table of contents is a special slide, having `id="toc"`.
3. The entries are grouped by means of an event-link in a text block:

```
<text1>
  <p><br/>
    <a href="event:goto#intro">Introduction</a>
  </p>
  ...
</text1>
```

Notes

1. Xpos're supports only end notes.
2. The use of notes is optional.
3. If used, the notes are to be listed on a special slide using the `<note>` tag.
4. Each note must have a unique [id](#). The id value may be a mnemonic key word.

```
<note id="odio">
  <p>[1] Odio montes neque. Luctus placerat blandit metus nisl orci eget tortor leo.
  Donec augue tellus sed dignissim libero.</p>
</note>
```

5. In the publication text event-links are used to refer to a note:

```
<a href="event:note#odio" class="note">[1]</a>
```

The class "note" makes the in-line reference red (see the CSS). Numbering the note references is left to the author.

Bibliography

1. The use of a bibliography is optional.
2. If used it must be on a special slide, just like the end notes.
3. A literature item is embedded in a `<ref>` tag and must have a unique [id](#). The id value may be combination of an author name, keyword from the title plus year, or any other combination that complies with the requirements of an ID.

```
<ref id="Barish2009">
  <p> Barish, S. and Daley, E. (2009) "Multimedia Scholarship for the Twenty-First
Century",
    <a href="http://net.educause.edu/ir/library/pdf/ffpiu047.pdf"
target="_blank">
      http://net.educause.edu/ir/library/pdf/ffpiu047.pdf
    </a>
  </p>
</ref>
```

4. In the publication text (and also in notes) event-links are used to refer to a literature item:

```
<a href="event:ref#Barish2009" class="ref">[Barish et al. 2009]</a>
```


Conversion

[From Word to XML](#)

[From Flash to HTML](#)

From Word to XML

Usually an author will start from an existing text. The XML text may be created by copying and pasting paragraph by paragraph, but this method is error prone if the source text contains a lot of mark-up (bold, italic, etc.). In that case conversion from the Word document to HTML and from HTML to XML may be more safe. However, the Word HTML text must be cleaned.

Do the following:

1. Save the document in Word as a filtered Web page (HTML). This already removes some superfluous code.
2. Go to the website Wordoff (<http://wordoff.org/>).
3. Copy and paste the filtered HTML document into the Wordoff window.
4. Click on button 'Clean up'.
5. Copy the result text into your XML editor
6. Edit and validate the text, and work it up to a valid Xpos're text.

There are, of course, other cleaning programs which can be used as well.

Created with the Personal Edition of HelpNDoc: [Easily create PDF Help documents](#)

From Flash to HTML

[As said before](#), the textbrowser can generate two flavors of HTML text: a paged version using slides and mimicking the Flash version, and a plain HTML version suitable for printing the entire document and for making an e-book.

Do the following:

1. Open textbrowser and make sure that you are in authoring mode (see ['Preferences'](#)).
2. Click on button 'HTML Slides' or 'HTML Plain'.
3. Copy the HTML output from the pop-up window into a WYSIWYG HTML-editor.
4. HTML Plain: you may want to beautify some parts of the HTML text because of differences in rendering between Flash and a regular web browser. This text relies on `RIP-HTML/HTML-plain.css` for its style; this CSS can of course be customized.
5. HTML Slides: you will not see much in the WYSIWYG editor, because this version relies on hidden div's which are made visible when the navigation button is clicked. So, switch to code view if you want to inspect the text. Be careful with any changes. This version uses `RIP-HTML/RIP-HTML.css`, which may be customized (colors and fonts).

Appendixes

[FAQ](#)

[Common errors](#)

[HTML tags supported](#)

[CSS properties supported](#)

[Textbrowser DTD](#)

[Publication DTD](#)

FAQ

- **Why can I not use the browser's back and forward button to navigate through the slides?**

The browser's back and forward button are used to navigate through web pages. The document reader is an application embedded in a single web page and, therefore, requires its own navigation buttons.

- **The size of the document reader is too big for the screen of my small laptop; can I resize it?**

1. Consider closing toolbars in order to free some screen space.
2. Switch to full screen mode, currently supported by most browsers.

- **Can I resize the text font?**

The text font cannot be resized by the user. The size is specified by the author in the CSS-file. However, you can use the browser's zoom facility to enlarge the text.

- **How can I print the enhanced publication?**

1. You may print individual slides if you want.
2. To print the complete text, use the plain HTML version.

- **How can I make foot notes or end notes?**

The document reader cannot display foot notes, only end notes. See topic [Special text](#) for details.

- **How can I add literature references?**

A literature list is a regular slide; text1 (and, if two columns are needed, text2) can be used for this purpose. See topic [Special text](#) for details.

To group several references within one pair of brackets, use a span-tag of class 'ref' to color the entire set of references.

Example:

```
<span class="ref">
[
<a class="ref" href="event:ref#Berents1991">Berents 1991,</a>
<a class="ref" href="event:ref#Berents1984">1984;</a>
<a class="ref" href="event:ref#Caenegem1954">Van Caenegem 1954</a>
]
</span>
```

This will look like: [**Berents 1991, 1984; Van Caenegem 1954**]

- **How to display a larger version of an image?**

The screen space for illustrations is limited. To see more details a larger version may be useful. A simple way is inserting a regular hyperlink into the caption of the smaller visual; the link text may be

'[large]' or simply a symbol '[+]'. Do not use an inline picture (like a magnifier) as an icon; such an image is not very well handled in captions because of the scaling. Alternatively, you may use an extension, such as the Image viewer (see simple demo). If you use a regular hyperlink to an image, do not forget to add the relative path (e.g. `visuals/myImage.jpg`).

- **How can a link be visible but not clickable?**

Check if text2 is not on top of text1 (and thus covering the link), or one of the text blocks is shielding the caption of the visual, which contains the link. The text blocks have a transparent background and may be larger than the text displayed. Click on the 'Format'-button and check size and position. Check the syntax of the hyperlink as well.

- **How to link to the enhanced publication?**

You can link from any web page to the enhanced publication just as linking to any other web document.

- **Can I link to a specific slide?**

The document reader can skip the start-up dialog and directly come up with a specific slide (**deep linking**). To get the right URL, navigate to that slide and click on button Preferences. The link to that particular slide is displayed in the pop-up window

- **Can a slide have a background image?**

To enliven slides with much text (for example two columns) may have a background image. However, a background image precludes other visuals on the slide. Moreover, because you need a rather large image, it will increase the preloading time. Use a preprocessed image with dim colors (for example grey tones or sepia) and position that behind the text. Both, text1 and text2 have a Z-index greater than the visuals.

- **On my Mac, linking to a specific PDF-page does not work in Firefox; what to do?**

On a Mac, Firefox may download the PDF in stead of opening it, or, if you have installed a less advanced plug-in, it will not jump to the right page. Use Safari in that case.

Common errors

Error messages

In most cases, an error message is displayed when the author makes mistakes in the authoring process, for example when:

- The XML-text is not well formed
- A visual file can not be found
- An internal link has not the appropriate syntax.

Not all user made errors can be trapped. As a rule of the thumb: if something unusual happens, check the XML code first by validating it against the related DTD.

The following error-situations may occur:

1. **Textbrowser.html** opens with a complete blank screen
 - **Textbrowser.swf** may be missing.
2. Only the welcome screen is displayed, no preloading 0%
 - Some file may be missing, for example **RIP-Flash.css**
3. Loading starts, but the percentage counter stops somewhere below 100%: **Error #2036 (preloading)**
 - One or more visuals are missing or filenames are misspelled. Filenames are case-sensitive.
4. Clicking on an internal link to an other slide does not produce the right result:
 - The event-protocol was not used
 - 'event:' was not followed by 'goto' or by an incorrect id.
5. Clicking on an internal link to a note or to a literature reference does not produce the right result:
 - Check the event link
 - Check the target (note, literature reference)
6. A link to a note or to a literature reference has no distinct color (e.g. red):
 - The class-attribute is missing in the <a>-tag.
 - The class is not defined in the CSS-file.
7. The 'Table of Contents' button is disabled:
 - There is no slide with id="toc".
8. Deep linking - wrong slide or file not found:
 - You may not have copied the entire URL as displayed in the Preference window.

HTML tags supported

Flash supports only a **subset of the HTML tags** and these tags are also recognized in an XML text. In addition it comes with a non-HTML-tag: <textformat>.

| Tag | Attributes | Remarks |
|--------------|--|---|
| <a> | href | <ul style="list-style-type: none"> External link: an absolute or relative URL Internal link: "event:verb#object#parameter" |
| | target | _blank, _parent, _top, _self |
| | | Renders text as bold. |
| | | Line break. Because the text is XML, can not be used. |
| | color | Only hexadecimal color (#FFFFFF) values are supported. |
| | face | Specifies the name of the font to use. You can specify a list of comma-delimited font names, in which case Flash Player selects the first available font. If the specified font is not installed on the user's computer system or isn't embedded in the SWF file, Flash Player selects a substitute font. |
| | size | Specifies the size of the font. You can use absolute pixel sizes, such as 16 or 18, or relative point sizes, such as +2 or -4. |
| | src | Specifies the URL to an image or SWF file. This attribute is required; all other attributes are optional. Formats: JPEG, GIF, PNG, and SWF files. |
| | width | The width of the image, SWF file in pixels. |
| | height | The height of the image, SWF file in pixels. |
| | align | Specifies the horizontal alignment of the embedded image within the text field. Valid values are left and right. The default value is left. |
| | hspace | Specifies the amount of horizontal space that surrounds the image where no text appears. The default value is 8. |
| | vspace | Specifies the amount of vertical space that surrounds the image where no text appears. The default value is 8. |
| | <i> | The <i> tag displays the tagged text in italics. An italic typeface must be available for the font used. |
| | This tag places a bullet in front of the text that it encloses. Note: Because Flash Player does not recognize ordered and unordered list tags (and), they do not modify how your list is rendered. All lists are unordered and all list items use bullets. | |
| <p> | align | align: Specifies alignment of text within the paragraph; valid values are: left, right, justify, and center. |
| | class | Specifies a CSS style class defined in the CSS-file. |
| | class | The tag is available only for use with CSS text styles. It supports the class attribute. |
| <textformat> | blockindent | Specifies the block indentation in points. |
| | indent | Specifies the indentation from the left margin to the first character in the paragraph. Both positive and negative numbers are acceptable. |
| | leading | Specifies the amount of leading (vertical space) between lines. Both positive and negative numbers are acceptable. |

| Tag | Attributes | Remarks |
|------------------|-------------|--|
| | leftmargin | Specifies the left margin of the paragraph, in points. |
| | rightmargin | Specifies the right margin of the paragraph, in points. |
| | tabstops | Specifies custom tab stops as an array of non-negative integers; |
| <u> | | Underlines a text. |

The following **HTML entities** are recognized:

| | |
|--------|---|
| < | < |
| > | > |
| & | & |
| " | " |
| ' | ' |

CSS properties supported

Flash supports the following CSS properties:

| Property | Usage and supported values |
|-----------------------|--|
| color | Only hexadecimal color values are supported. Named colors (such as blue) are not supported by Flash, but can be used in tags of the document reader. Colors in the CSS are written in the following format: #FF0000. |
| display | Supported values are: <ul style="list-style-type: none">• inline• block• none |
| font-family | A comma-separated list of fonts to use, in descending order of desirability. Any font family name can be used. If you specify a generic font name, it is converted to an appropriate device font. The following font conversions are available: <ul style="list-style-type: none">• mono is converted to typewriter• sans-serif is converted to _sans• serif is converted to _serif. |
| font-size | Only the numeric part of the value is used. Units (px, pt) are not parsed; pixels and points are equivalent. |
| font-style | Recognized values are: <ul style="list-style-type: none">• normal• italic |
| font-weight | Recognized values are: <ul style="list-style-type: none">• normal• bold |
| Kerning | Recognized values are true and false. Kerning is supported for embedded fonts only. Certain fonts, such as Courier New, do not support kerning. The kerning property is only supported in SWF files created in Windows, not in SWF files created on the Macintosh. However, these SWF files can be played in non-Windows versions of Flash Player and the kerning still applies. |
| leading | The amount of space that is uniformly distributed between lines. The value specifies the number of pixels that are added after each line. A negative value condenses the space between lines. Only the numeric part of the value is used. Units (px, pt) are not parsed; pixels and points are equivalent. |
| letter-spacing | The amount of space that is uniformly distributed between characters. The value specifies the number of pixels that are added after each character. A negative value condenses the space between characters. Only the numeric part of the value is used. Units (px, pt) are not parsed; pixels and points are equivalent. |
| margin-left | Only the numeric part of the value is used. Units (px, pt) are not parsed; pixels and points are equivalent. |
| margin-right | Only the numeric part of the value is used. Units (px, pt) are not parsed; pixels and points are equivalent. |
| text-align | Recognized values are: <ul style="list-style-type: none">• left• center• right• justify. |

| Property | Usage and supported values |
|------------------------|--|
| text-decoration | Recognized values are: <ul style="list-style-type: none">• none• underline. |
| text-indent | Only the numeric part of the value is used. Units (px, pt) are not parsed; pixels and points are equivalent. |

Textbrowser DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XPOS'RE 2.0 - 29 March 2012
== DTD for TEXTBROWSER.HTML ==
Unless otherwise indicated files are expected in the same directory as Textbrowser
itself.
-->

<!--
=====
          I. APPLICATION DATA
=====
-->

<!ELEMENT appData (mode, urlSite?, urlText, urlPDF?, datePDF?, urlCSS, imagePath,
welcomeText,
scrollDelta, measuringHelp, useOverlay, templates, htmlTemplates)>

<!-- Mode of Textbrowser: effects default buttonbar (to be changed in Preferences:
1. authoring: all buttons
2. reading: only buttons required for reading are visible
3. protected: no switch to authoring mode through buttonbar
-->
<!ELEMENT mode (#PCDATA) >

<!-- Url of the website where the text will be published -->
<!ELEMENT urlSite (#PCDATA) >

<!-- File name of publication text -->
<!ELEMENT urlText (#PCDATA) >

<!-- Optional url of PDF version of the publication (not necessarily in same
directory) -->
<!ELEMENT urlPDF (#PCDATA) >

<!-- Optional date of the PDF file -->
<!ELEMENT datePDF (#PCDATA) >

<!-- Cascading stylesheet -->
<!ELEMENT urlCSS (#PCDATA) >

<!-- Subdirectory where the images are; include '/' at the end -->
<!ELEMENT imagePath (#PCDATA) >

<!-- Text displayed at start-up during preloading -->
<!ELEMENT welcomeText (p+) >

<!-- Number of lines scrolled in a single step by scroll buttons; default: 10 -->
<!ELEMENT scrollDelta (#PCDATA)>

<!--Help text on | off when measuring tool is displayed -->
<!ELEMENT measuringHelp (#PCDATA) >

<!-- Use overlay on | off with: href="event:overlay#title#url" -->
<!ELEMENT useOverlay (#PCDATA) >

<!--
=====
          II. AUTHORING TEMPLATES
=====
-->
<!ELEMENT templates (template)+ >
```

```

<!ELEMENT template (text1?, text2?, visuals?)>
<!ATTLIST template
    id ID #REQUIRED>

<!ELEMENT text1 (#PCDATA) >
<!ATTLIST text1
    x CDATA #IMPLIED
    y CDATA #IMPLIED
    width CDATA #IMPLIED
    height CDATA #IMPLIED>

<!ELEMENT text2 (#PCDATA) >
<!ATTLIST text2
    x CDATA #IMPLIED
    y CDATA #IMPLIED
    width CDATA #IMPLIED
    height CDATA #IMPLIED>

<!ELEMENT visuals (#PCDATA) >
<!ATTLIST visuals
    x CDATA #IMPLIED
    y CDATA #IMPLIED
    size CDATA #IMPLIED
    orientation CDATA #IMPLIED> <!-- horizontal | vertical -->

<!--
=====
        III. CONVERSION TEMPLATES
=====
-->
<!ELEMENT htmlTemplates (htmlTemplate)+ >
<!ELEMENT htmlTemplate (html)>
<!ATTLIST htmlTemplate
    id ID #REQUIRED>

<!ELEMENT html (head, body) >
<!ELEMENT head (title, (script | link)*) >
<!ELEMENT title (#PCDATA) >

<!ELEMENT script (#PCDATA)>
<!ATTLIST script
    type CDATA #REQUIRED
    src CDATA #IMPLIED>

<!ELEMENT link EMPTY >
<!ATTLIST link
    href CDATA #REQUIRED
    rel CDATA #IMPLIED
    type CDATA #IMPLIED
    media CDATA #IMPLIED >

<!ELEMENT body (#PCDATA | map | div | button)*>
<!ATTLIST body
    onload CDATA #IMPLIED
    onkeydown CDATA #IMPLIED>

<!ELEMENT map (area)+ >
<!ATTLIST map
    name CDATA #REQUIRED>

<!ELEMENT area EMPTY>
<!ATTLIST area
    href CDATA #REQUIRED

```

```

    shape CDATA #REQUIRED
    coords CDATA #REQUIRED
    title CDATA #REQUIRED>

<!ELEMENT div (#PCDATA | p | img | div | select | button | a | h1 | small)*>
<!ATTLIST div
    id CDATA #REQUIRED
    class CDATA #IMPLIED>

<!ELEMENT select (#PCDATA) >
<!ATTLIST select
    id ID #REQUIRED
    onchange CDATA #IMPLIED>

<!--
=====
    HTML AND FLASH TAGS
    as shared by all sections above
=====
-->
<!ELEMENT h1 (#PCDATA | small | br)* >
<!ELEMENT h2 (#PCDATA | small | br)* >
<!ELEMENT h3 (#PCDATA | small | br)* >

<!ELEMENT p (#PCDATA | b | u | i | br | font | span | textformat | input)*>

<!ATTLIST p
    align (left | right | justify | center) #IMPLIED
    class CDATA #IMPLIED>

<!ELEMENT b (#PCDATA | i | u)* >
<!ELEMENT i (#PCDATA | b | u)* >
<!ELEMENT u (#PCDATA | b | i)*>
<!ELEMENT br EMPTY >

<!ELEMENT font (#PCDATA)>
<!ATTLIST font
    color CDATA #IMPLIED
    face CDATA #IMPLIED
    size CDATA #IMPLIED>

<!ELEMENT span (#PCDATA | b | u | i | br)*>
<!ATTLIST span
    class CDATA #REQUIRED>

<!ELEMENT a (#PCDATA | br | i | b | u | img)*>
<!ATTLIST a
    href CDATA #REQUIRED
    class CDATA #IMPLIED
    target CDATA #IMPLIED>

<!ELEMENT img EMPTY >
<!ATTLIST img
    id ID #IMPLIED
    src CDATA #REQUIRED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    vspace CDATA #IMPLIED
    hspace CDATA #IMPLIED
    align CDATA #IMPLIED
    usemap CDATA #IMPLIED
    border CDATA #IMPLIED
    title CDATA #IMPLIED>

```

```
<!ELEMENT small (#PCDATA) >

<!ELEMENT button (#PCDATA)>
<!ATTLIST button
  id ID #REQUIRED
  onclick CDATA #REQUIRED
  title CDATA #IMPLIED>

<!ELEMENT input (#PCDATA) >
<!ATTLIST input
  type CDATA #REQUIRED
  id ID #REQUIRED>

<!-- Proprietary Flash tags -->
<!ELEMENT textformat EMPTY>
<!ATTLIST textformat
  blockindent CDATA #IMPLIED
  indent CDATA #IMPLIED
  leading CDATA #IMPLIED
  leftmargin CDATA #IMPLIED
  rightmargin CDATA #IMPLIED
  tabstops CDATA #IMPLIED>
```

Publication DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XPOS'RE 2.0 - 29 March 2012
== DTD for XPOS'RE PUBLICATIONS ==
Note:
- attribute values of x, y, width, height, size in pixels
- color: either #code or web named color
-->

<!-- ===== -->
<!-- Entities -->
<!ENTITY % textContent "(p | ul | h1 | h2 | h3 | h4 | h5 | h6 | br | textformat)*"
>
<!ENTITY % pContent "(#PCDATA | b | i | u | a | img | br | span | textformat | font)*">
<!ENTITY % capContent "(#PCDATA | b | i | u | a | br | span | textformat | font)*"
>
<!ENTITY % size "x?, y?, width?, height?" >
<!ENTITY % size_visual "x?, y?, width?" >
<!-- ===== -->

<!-- Publication document structure -->
<!ELEMENT doc (meta, slides) >

<!-- Metadata -->
<!ELEMENT meta (mainTitle, author, creator?, issued?, modified?, description?)>
<!ELEMENT mainTitle (#PCDATA) > <!-- publication title -->
<!ELEMENT author (#PCDATA) > <!-- first name (or initials) and last name -->
<!ELEMENT creator (#PCDATA) > <!-- e.g. the editor of the text -->
<!ELEMENT issued (#PCDATA) > <!-- date; any format allowed -->
<!ELEMENT modified (#PCDATA) > <!-- date; any format allowed -->
<!ELEMENT description (%textContent;) >

<!-- Slides -->
<!ELEMENT slides (slide+)>
<!ELEMENT slide (pdfPage?, title, visual*, button*, text1?, text2?, more?)>
<!ATTLIST slide
    id ID #REQUIRED
    template CDATA #IMPLIED<!-- id of one of the templates defined in
Textbrowser.xml -->

<!ELEMENT pdfPage (#PCDATA) ><!-- number of the PDF page related to the current slide
-->
<!ELEMENT title %pContent; ><!-- slide title -->

<!-- Visuals -->
<!ELEMENT visual (url, onClick?, cap?, creator?) ><!-- attributes only required if
format different from template -->
<!ATTLIST visual
    x CDATA #IMPLIED
    y CDATA #IMPLIED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    border CDATA #IMPLIED<!-- true | false -->

<!ELEMENT url (#PCDATA) ><!-- url of the image -->
<!ELEMENT onClick EMPTY>
<!ATTLIST onClick
    href CDATA #REQUIRED<!-- typically an event-link -->
<!ELEMENT cap %capContent;>

<!-- Buttons -->
<!ELEMENT button EMPTY>
```

```

<!ATTLIST button
  href CDATA #REQUIRED
  label CDATA #REQUIRED
  x CDATA #REQUIRED
  y CDATA #REQUIRED
  width CDATA #REQUIRED>

<!-- Texts -->
<!ELEMENT text1 (%textContent; | note | ref)*>
<!ATTLIST text1
  x CDATA #IMPLIED
  y CDATA #IMPLIED
  width CDATA #IMPLIED
  height CDATA #IMPLIED
  borderColor CDATA #IMPLIED
  backgroundColor CDATA #IMPLIED>

<!ELEMENT text2 (%textContent; | note | ref)*>
<!ATTLIST text2
  x CDATA #IMPLIED
  y CDATA #IMPLIED
  width CDATA #IMPLIED
  height CDATA #IMPLIED
  borderColor CDATA #IMPLIED
  backgroundColor CDATA #IMPLIED>

<!ELEMENT ref %textContent; ><!-- a bibliographic entry referred to through <a
class="ref" href="event:ref#..."-->
<!ATTLIST ref
  id ID #REQUIRED>

<!-- HTML tags -->
<!ELEMENT h1 %pContent;>
<!ELEMENT h2 %pContent;>
<!ELEMENT h3 %pContent;>
<!ELEMENT h4 %pContent;>
<!ELEMENT h5 %pContent;>
<!ELEMENT h6 %pContent;>

<!ELEMENT p %pContent; >
<!ATTLIST p
  align (left | right | justify | center) #IMPLIED
  class CDATA #IMPLIED>

<!ELEMENT b (#PCDATA | i | u | a | font)* >
<!ELEMENT i (#PCDATA | b | u | a | font)* >
<!ELEMENT u (#PCDATA | b | i | font)*>
<!ELEMENT br EMPTY >
<!ATTLIST br
  class CDATA #IMPLIED>
<!ELEMENT ul (li, br?)+ ><!-- ordered lists are not supported by Flash -->
<!ELEMENT li %pContent; >

<!ELEMENT font (#PCDATA)>
<!ATTLIST font
  color CDATA #IMPLIED
  face CDATA #IMPLIED
  size CDATA #IMPLIED>

<!ELEMENT a (#PCDATA | br | i | b | u | img)*>
<!ATTLIST a
  href CDATA #REQUIRED
  class CDATA #IMPLIED

```

```

        target CDATA #IMPLIED>

<!ELEMENT img EMPTY ><!-- image embedded in the text -->
<!ATTLIST img
    src CDATA #REQUIRED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    vspace CDATA #IMPLIED
    hspace CDATA #IMPLIED
    align CDATA #IMPLIED>

<!ELEMENT span %pContent;>
<!ATTLIST span
    class CDATA #REQUIRED>

<!-- Proprietary Flash tags -->
<!ELEMENT textformat %pContent;>
<!ATTLIST textformat
    blockindent CDATA #IMPLIED
    indent CDATA #IMPLIED
    leading CDATA #IMPLIED
    leftmargin CDATA #IMPLIED
    rightmargin CDATA #IMPLIED
    tabstops CDATA #IMPLIED>

<!-- Special texts -->
<!ELEMENT more %textContent; ><!-- continuation text, referred to through <a
href="event:more">More...</a> -->

<!ELEMENT note %textContent; ><!-- (foot)note referred to through <a class="note"
href="event:note#...-->
<!ATTLIST note
    id ID #REQUIRED>

```