Mate in 3D – Publishing Interactive Content in PDF3D

Frank Breuel
Universität Rostock, Germany and Fraunhofer Austria. Email: frank.breuel@uni-rostock.de

René Berndt
Graz University of Technology, Austria. Email: r.berndt@cgv.tugraz.at

Torsten Ullrich
Fraunhofer Austria Research GmbH, Graz, Austria. Email: torsten.ullrich@fraunhofer.at

Eva Eggeling
Fraunhofer Austria Research GmbH, Graz, Austria. Email: eva.eggeling@fraunhofer.at

Dieter W. Fellner
Fraunhofer IGD and TU Darmstadt, Germany. Email: d.fellner@igd.fraunhofer.de

Abstract: *In this paper we describe a pipeline for publishing interactive multimedia content. The Portable Document Format (PDF) offers the possibility to include 3D visualizations, textual representation and interactivity (via scripting technology) in one multimedia container, which will be the upcoming standard for multimedia long-term archiving. Our system demonstrates its potential for surplus value eBooks. By the example of chess we developed a publishing pipeline to create interactive books. Usually, chess games and positions are recorded using the algebraic chess notation, which is mainly an annotated list of moves. In combination with a time-dependent 3D visualization, each move corresponds to a specific game position. This correspondence is encoded in hyperlinks from the textual representation to the 3D visualization. This linkage improves readability and usability of chess notations significantly. Furthermore, using an established file format our eBooks can be opened by any compliant PDF viewer.*

Keywords: *PDF3D; chess; multimedia; interactive content.*

Introduction

According to José Raúl Capablanca[1] -a great world chess master- "Chess books should be used as we use glasses: to assist the sight, although some players make use of them as if they conferred sight" (Evans & Wiswell, 1953). Chess, the popular board game, has been fascinating its players for centuries. Due to its tactical character, the analysis of ancient games of chess is an important aspect in chess theory. Many games are recorded, evaluated and analyzed by interested players. Usually, chess games and positions are recorded using the algebraic chess notation (FIDE, 2008), which is mainly an annotated list of moves. In order to analyze a game, one has to take a chess board and chess players and replay the game move by move; especially the analysis of variants and alternatives can be cumbersome.

The success story of information technology in the field of chess applications comprehends an algorithmic breakthrough best described by two quotes of Garry Kasparov[2], a famous world chess champion: "Inevitably the machines must win, but there is still a long way to go before a human on his or her best day is unable to defeat the best computer."[3] Only a few years later he said that "ironically, the main task of chess software companies today is to find ways to make the program weaker, not stronger, and to provide enough options so that any user can pick from different levels

[1] http://en.wikipedia.org/wiki/Jos%C3%A9_Ra%C3%BAl_Capablanca
[2] http://en.wikipedia.org/wiki/Kasparov
[3] http://www.chessquotes.com/player-kasparov

and the machine will try to make enough mistakes to give him a chance."[3] Despite this algorithmic breakthrough, the questions of usability and long-term archiving have been unanswered so far.

In this paper we present a solution for digital publication and archiving of chess games based on established PDF standards. The resulting documents can be read with any PDF viewer. Our system is capable of generating an interactive, self-contained PDF document automatically from a given portable game notation (PGN) file (Edwards, 1994). The PGN file format is the most popular way to store chess games digitally. Thousands of played games in different varieties have been made available up to now in this format.

PDF is the perfect choice for publishing multimedia eBooks. The next version PDF/A-2 (ISO 2008) of the archival standard PDF/A (ISO, 2005) based on PDF 1.7 is expected to contain support for 3D. Consequently it becomes possible to generate state-of-the-art multimedia content that can be used for long-time archiving and storage. Furthermore, chess games that were created with our software can thus be displayed at any time by any PDF viewer. Using digital library techniques, every game of chess can be identified uniquely. Embedded into a standardized document, a chess game can be identified via the document's digital object identifier (DOI) and thus be referenced.

Related Work

Concerning chess books several techniques for publication are established. The most commonly used chess books (tournament publications as well as textbooks) are published the classical way (Appelt, 1988).

With upcoming information technology systems, a growing number of chess publications have been digitally born. These systems can be distinguished from each other by the visualization technique used. On the one hand, there are a large number of software packages that need to be installed locally. On the other hand, online solutions on various websites allow a user to recapitulate a chess game using advanced web technology, namely JavaScript and HTML.

While there are a variety of eBooks formats[4] available, only a few offer support for multimedia content. This is not surprising since the "classic" eBooks formats mostly target hardware devices, which are not designed for power-consuming multimedia.

Software products that have to be installed locally, such as *Chess Base Light,* offer a range of options for visualizing and editing chess games. Such programs usually supply a big chess database including various matches. In addition, they offer the possibility to recapitulate a chess match graphically with included annotations, comments, etc. Furthermore, it is possible to create and save new games. Nevertheless, these applications are highly dependent on the operating system, cannot be used on eBook devices and are not a serious option for long-term archiving.

The online approaches via current web technology show an uneven picture. Some solutions need permanent online access, some have limited visualization capabilities, some are "optimized" for special browsers and environments, and none of them is a serious alternative for long-term archiving. The new features of HTML5 (W3C, 2011), like the "canvas" element for 2D drawings, promise a new and cross-platform solution for creating and presenting interactive content. Even 3D support will be available through WebGL (Khronos Group, 2011). Currently only Mozilla Firefox and Chrome (and the beta releases of Safari and Opera) offer support for WebGL. HTML5 will become a W3C recommendation in 2014.

Technology

PDF 3D

With the introduction of PDF 1.6 (Adobe Acrobat 7) it is possible to add 3D content to a PDF document. The 3D content – also referred to as 3D annotations – is defined within a rectangular area of the page, which optionally can be overlaid with a bitmap for non-3D capable PDF viewers. Basic features of 3D annotations include multiple pre-defined views, various render modes and different navigation modes.

[4] http://en.wikipedia.org/wiki/Comparison_of_e-book_formats

A crucial aspect of the interactive 3D annotation is the definition of the behavior for the scene. For this purpose Adobe introduced the 3D JavaScript API (Adobe, 2007). This API is separate from the document-level JavaScript API for manipulating the "classical" PDF content. This API provides special methods and classes to realize 3D programming, e.g., *Node*, *Light*, *Material* or *Camera*. Callback classes for various events can be used to define a custom interaction for the 3D scene, e.g. mouse events, keyboard events, timer, etc.

According to the standard by Adobe there is support by two formats for representing 3D content in a PDF document. These formats are the RPC file format (Adobe, 2008) and the Universal 3D File Format (U3D) (ECMA International, 2007).

XSL-FO (XSL-Formatting Objects)

XSL Formatting Objects (XSL-FO) is an XML-based markup language for formatting and presentation of XML-documents (W3C, 2006). It defines a set of elements and attributes for describing the visual appearance of XML-encoded data.

The general idea behind XSL-FO is to transform an XML document containing data into another XML document containing the data and the XSL-FO elements for the visual presentation. This document is then converted by an FO processor (FOP) to a given target format, which has to be printable, writeable or even both. XSL-FO is designed to support various render targets depending on the implementation of the FO processor. Nevertheless, in most cases the render target is PDF.

FO3D (Formatting Objects 3D)

FO3D is an extension to the XSL Formatting Objects (XSL-FO) standard (Buchgraber, Berndt, Havemann, & Fellner, 2010). Since the XSL-FO is designed for rendering two dimensional layouts, the FO vocabulary lacks support for embedding 3D content. However, support for arbitrary objects is already included within the XSL standard.

Figure 1 shows the definition of a 3D object using FO3D. The root element of the FO3D object with 3D content is *object-3d*. The following sub-tree contains information about additional data used for creating the 3D annotation, like additional resources (e.g., other 3D models, images) or JavaScript code.

FO3D utilizes the *fo:instream-foreign-object* element, which "is used for an inline graphic or other 'generic' object where the object data resides as descendants of the *fo:instream-foreign-object*, typically as an XML element sub tree in a non-XSL namespace" (W3C 2006). FO3D primarily targets the integration of 3D content and related metadata in PDF documents; therefore the proposed FO3D vocabulary contains various settings and concepts following the ISO-standardized PDF-1.7 specification (ISO-32000-1 2008). A detailed description of the FO3D vocabulary can be found in Buchgraber et al. (2010). Figure 2 shows the typical workflow for XSL-FO processing with FO3D. FO3D has been implemented for Apache FOP (Apache, 2011) and is available for download at http://fo3d.sourceforge.net.

```
1   <?XML version="1.0" encoding="UTF-8"?>
2   <fo:root XMLns:fo=http://www.w3.org/1999/XSL/Format
    XMLns:fox="http://XMLgraphics.apache.org/fop/extensions">
3   <fo:layout-master-set>
4   …
5   <fo:instream-foreign-object XMLns="http://fo3d.sourceforge.net/ns">
6      <object-3d src="./Fop files/Models/block.u3d" width="15cm" height="15cm"
        altsrc="./Fop files/Models/Overlay.jpg">
7         <default-view background-color="white" render-mode="solid" light-mode="cad">
8            <camera position="0,50,0" target="0,0,0" up="0,0,-1" />
9         </default-view>
10        <resources>
11           <resource src="./Fop files/Models/Chessfigures/u3d/pawnBlack.u3d"
              name="pawnBlack" />
12        </resources>
13        <scripts>
14           <script type="open" src="./Fop files/Sources/Board.js" />
15           …
16        </scripts>
17     </object-3d>
18  </fo:instream-foreign-object>
19  …
```

Figure 1. Embedded 3D content information in an FO file. The object-3d builds the root of a 3D annotation.

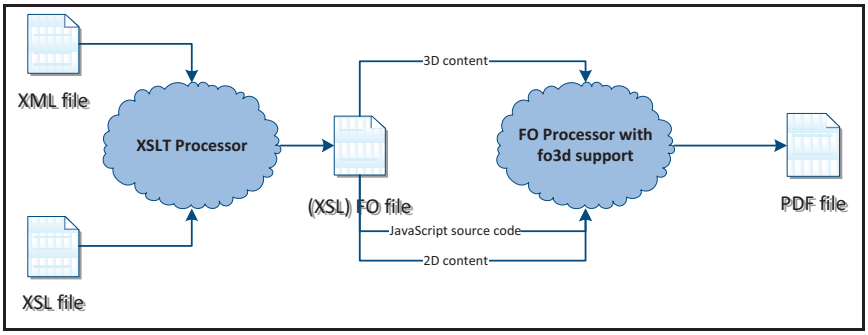

Figure 2. Workflow diagram of the XSL-FO formatting process, embedding 3D content in PDF documents (Buchgraber, et al., 2010)

Processing Pipeline

Our proposed system implements the complete processing pipeline starting from the source PGN to the final PDF document (see Figure 3). Our system consists of several parts that will be explained in more detail in the following paragraphs.

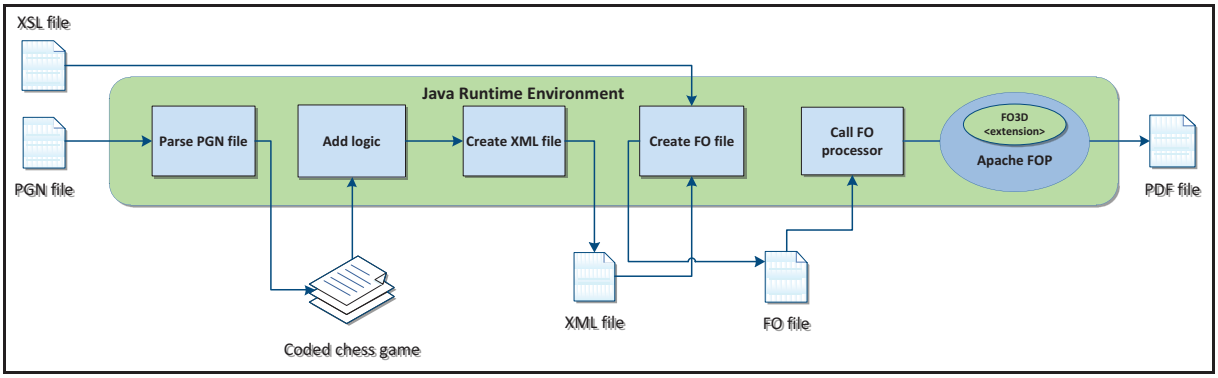

Figure 3. General structure of the processing pipeline

Parsing a PGN file

The first step of our pipeline is to parse the given PGN file. The PGN standard was developed and published by Steven J. Edwards in 1994 (Edwards, 1994). It stores a game notation as plain text, which can be very easily read and written by humans (Figure 4). Also, most chess programs, e.g. ChessBase[5], can export their data to PGN. The first part of a PGN file consists of a set of tuples containing general metadata of the match such as date, player information, location, etc. enclosed in square brackets.

```
[Event "New York Rosenwald"]
[Site "New York"]
[Date "1956.10.07"]
[White "Byrne,Donald"]
[Black "Fischer,Robert James"]
[Result "0-1"]
[Eco "D97"]

1.Nf3 {A noncommittal move. From here, the game can develop into a number of different
openings.} Nf6 2.c4 g6 3.Nc3 Bg7 4.d4 O-O {Fischer castles, concentrating on protecting his
king immediately.} 5.Bf4 d5 6.Qb3 {The so-called Russian System, putting pressure on
Fischer's central d5 pawn.} dxc4 7.Qxc4 c6 8.e4 Nbd7  9.Rd1 Nb6 10.Qc5 Bg4 {At this point,
Byrne's pieces are more developed, and he controls the center squares. However, Fischer's
king is well-protected, while Byrne's king is not.} 11.Bg5 (11. Be2 Nfd7 12. Qa3 Bxf3 13.
Bxf3 e5 14. dxe5 Qe8 15. Be2 Nxe5 16. O-O {and white is better.}) Na4 12.Qa3 {Here Fischer
cleverly offers up his Knight, but if Byrne takes it with} (12.Nxa4 {Fischer will play}
Nxe4 {, and Byrne then suddenly has some terrible choices:} 13. Qxe7 (13. Bxe7 Nxc5 14.
Bxd8 Nxa4 15. Bg5 Bxf3 16. gxf3 Nxb2 {gives Fischer an extra pawn and ruins Byrne's pawn
structure.})
…
```

Figure 4. PGN file including moves, comments and variants in the chess match

Next up follows the declaration of all moves made in that game. In addition, a move can contain comments (enclosed in curly brackets), NAGs (Numeric Annotation Glyphs) and alternative variants (enclosed in round brackets). NAGs (Edwards, 1994) are used to annotate chess games providing an assessment of a chess move or a chess position, e.g. good move, poor move, white has a moderate advantage, etc.

Our PGN parser builds up a tree structure based on half-moves. A half-move is defined as the move of one player (the half-move of white followed by the half-move of black is then a full move). Each node in the tree represents a half-move H_1. The next half-move H_2 is stored as the first child of the node containing H_1. Variants are stored as siblings to H_2. A match without variants would result in a linear graph. Each node contains several pieces of information about the half-move, like the color, the number of already made half-moves and the source and destination field of the piece.

Adding additional data

In most cases the abbreviated (or short) algebraic notation is used. It just records the moving piece along with the destination field. If this information does not unambiguously define the half-move, information about the source field is added, e.g. Nf3d or Qxe5. Here the first notation means that the *Knight* on field *f3* has to move to field *d*; the second signifies that the *Queen* beats the chess player located on field *e5*. There exist a lot of possibilities to quote a single chess move and all of them have to be parsed and checked.

Therefore the starting-field and the actually moved chess piece will be calculated for each half-move. If there are more than one or even no possible chess piece for that half-move notation, the semantic structure of the file seems to be incorrect so the file will be rejected. Otherwise, if exactly one chess piece was found, this information will be added to the node containing the half-move, and the position of all pieces will be updated. These calculations have to be performed to the main line and all variants.

[5] http://www.chessbase.com/

Generating the XML and (XSL) FO file

The complete parsed half-move tree is then exported as an XML document. This XML document is used for generating the 3D JavaScript code for that particular match.

Figure 5 shows the structure of the XML document. The first part contains the metadata of the match, as found in the PGN file. The match itself is listed as a sequence of half-moves. Every outlined half-move contains a unique identification number, a reference to the previous and to the next move.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <chessgames>
3   <chessgame>
4   <filename>gameOfTheCentury.pgn</filename>
5   <result>0-1</result>
6   <attributes>
7      <attribute>[Event "New York Rosenwald"]</attribute>
8      …
9   </attributes>
10  <moves>
11     <move>
12        <halfmoveWhite moveNumber="1" mainLine="true" color="white" moveFull="Nf3"
           id="0" idParent="-1" idChild="1">
13          <chessmove from="g1" to="f3" switch=" " />
14          <comment>A noncommittal move. From here, the game can develop into a
             number of different openings.</comment>
15        </halfmoveWhite>
16        <halfmoveBlack moveNumber="1" mainLine="true" color="black" moveFull="Nf6"
           id="1" idParent="0" idChild="2">
17          <chessmove from="g8" to="f6" switch=" " />
18        </halfmoveBlack>
19     </move>
20  <move>
21  …
```

Figure 5. XML document example containing the sequence of half-moves.

Currently there is no support for adding a 2D overlay to the 3D annotation, e.g. to display the comments for a particular half-move. Our solution is to use billboarding, i.e. rendering the text to a bitmap and including this bitmap as a resource within the 3D annotation. Figure 6 shows how the images for the comments are represented within the XML document.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <chessgames>
3      <chessgame>
4         …
5         <moves>
6         …
7         </moves>
8         <commentImages>
9            <commentImg commentId="commentImage0">./Fop
              files/Models/Images/commentImage_0.jpg</commentImg>
10           <commentImg commentId="commentImage5">./Fop
              files/Models/Images/commentImage_5.jpg</commentImg>
11           …
12        </commentImages>
13        <moveImages>
14           <moveImg moveId="moveImage0">./Fop files/Models/Images/moveImage_0.jpg
             </moveImg>
15           …
16        </moveImages>
17     </chessgame>
18  </chessgames>
```

Figure 6. Associated relative paths for each comment and move in the XML file.

The resulting XML document is then transformed using XSL into an FO document. Figure 7 shows how a half-move is laid out for the PDF document. Line 25 shows how the textural representation of the half-move can interact with the 3D annotation.

```
<?XML version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" XMLns:xsl="http://www.w3.org/1999/XSL/Transform">
…
<xsl:template match="move">
   <inline XMLns="http://www.w3.org/1999/XSL/Format">
      <xsl:if test="halfmoveWhite/@mainLine='true'">
         <inline text-decoration="underline">
            <xsl:if test="halfmoveWhite/@moveNumber != -1">
               <xsl:value-of select="halfmoveWhite/@moveNumber" />
               <xsl:text>. </xsl:text>
            </xsl:if>
         </inline>
      </xsl:if>
      <xsl:if test="halfmoveWhite/@mainLine='false'">
         <xsl:if test="halfmoveWhite/@moveNumber != -1">
            <xsl:value-of select="halfmoveWhite/@moveNumber" />
            <xsl:text>. </xsl:text>
         </xsl:if>
      </xsl:if>
   </inline>

   <xsl:variable name="numberWhite">
      <xsl:value-of select="halfmoveWhite/@id" />
   </xsl:variable>
   <xsl:variable name="javascriptWhite">
      <xsl:value-of select="concat('javascript:var
        h=getAnnots3D(0)[0];if(h.activated)h.context3D.goTo(',$numberWhite,')')" />
   </xsl:variable>
…
```

Figure 7. XSL document example, containing information on how to layout the chess data

Creation of the PDF document

The generated FO document is then processed by Apache FOP. In order to process the custom chess XML structure described above, FO3D has been extended using a custom FO3D extension. FO3D custom extensions are typically used for tasks which are not possible using XSLT and Javascript, e.g. generating images or loading arbitrary external files referenced from the XML input.

Figure 8 shows the use of the PGN extension within the FO document. The PGN extension generated the necessary JavaScript code, which is necessary for the animation of the chess match within the 3D annotation. It also provides a mechanism for the document level JavaScript to interact with the 3D annotation.

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo=http://www.w3.org/1999/XSL/Format
xmlns:fox="http://xmlgraphics.apache.org/fop/extensions">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="A4"
         …
            <extensions>
               <pgn src="./pgnAsXml.xml" />
            </extensions>
         </object-3d>
      </fo:instream-foreign-object>
   </fo:block>
   <fo:block font-family="Helvetica" font-size="11">
      <inline xmlns="http://www.w3.org/1999/XSL/Format">
         <inline text-decoration="underline">1. </inline>
      </inline>
      <inline font-style="italic" xmlns="http://www.w3.org/1999/XSL/Format">
         <basic-link external-destination="javascript:var
          h=getAnnots3D(0)[0];if(h.activated)h.context3D.goTo(0)" color="blue">
            <inline text-decoration="underline">Nf3</inline>
         </basic-link>
         …
```

Figure 8. Part of the FO file used by the FO processor with the PGN extension.

It is possible to jump to any chess move, to take a step forward or backwards as well as to display comments and the move in the scene itself. Besides the 3D annotation the game notation itself is also added to the PDF document in textual hyperlinks. A left click on such a hyperlink, which represents a

single chess half-move, triggers the 3D annotation to display the position after that particular half-move. Lines 16-18 in Figure 8 show how to create such a clickable hyperlink within the text part of the PDF document.

Figure 9. Selected views of the resulting PDF file to a given chess match. **Top left:** 3D annotation showing the position after Fischer's 3rd move. The comment is directly show within the 3D scene. **Top right:** The final position of the match. **Bottom:** The 3D annotation is displayed in a floating window beside the document allowing the viewer to scroll through the text while keeping the 3D annotation visible even on page breaks

Figure 9 shows a resulting PDF document. The user can interact with the 3D scene in several ways. Thus the scene itself can be displayed in a full-screen mode or as a floating window so the user can scroll through the document and always have a view of the match.

Comments and moves are displayed in the 3D annotation as well as in the text. Buttons allow easy navigation within the chess match to go to the next, previous, last, or to the first move in the match or even to start an automatic play-back of the match.

Furthermore it is possible, by clicking on the lower left button, to enable or to disable the visibility of all buttons and the text fields. This is needed in case the scene is located in a floating window and so small that otherwise the match actually is not comprehensible in a graphical way.

Conclusion and Further Work

In this article we have presented the steps and technologies to develop a complete 3D chess eBook using the PDF standard. The pipeline takes care of parsing PGN files (the most commonly-used chess game description file format), extracting content and creating files for the interfaces.

Although several software solutions are already available today, none of them offers or combines the benefits of modern eBook technology (concerning usability and mobility) with the requirements of a digital library (long-term archiving, indexing, referencing). Our solution is platform independent, as only a PDF reader is needed. Furthermore, PDF with 3D support is expected to become the next electronic document file format for long-term preservation (PDF/A-2). As a consequence, our solution is the perfect basis for archiving chess games. In combination with digital object identifier (DOI) techniques a unique identification of each match is possible.

Concerning future work, two important aspects have to be considered. The current solution is optimized for chess games, and meets the special requirements of this field of application. In the future a more general approach will regard a chess game as a specialization of a process. Process documentation including its visualization, metadata (information about the data processed), and paradata (information about the process itself) is an upcoming challenge in a vast range of information technology applications: from civil engineering to medical care via electronic government.

While the first important aspect generalizes the field of application, the second one specializes the resulting documentation format. Different fields of application require adaptations to systems already in use. As our pipeline has a high degree of flexibility and easy expandability, our solution is prepared for cross-media publishing. Consequently, integration tasks should not pose any problems.

Acknowledgments

The work presented in this paper was partially supported by the German Research Foundation DFG under grant INST 9055/1-1 (PROBADO project http://www.probado.de/).

References

Adobe. (2007). *JavaScript™ for Acrobat® 3D Annotations API Reference*. Retrieved April 4, 2011 from http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/js_3d_api_reference.pdf.

Adobe. (2008). *PRC file format specification - Version 8137*. Retrieved April 4, 2011 from http://livedocs.adobe.com.

Apache Software Foundation. (2011). *Formatting Objects Processor – Apache FOP*. Retrieved April 4, 2011 from http://XMLgraphics.apache.org/fop/.

Appelt, W. (1988). Typesetting chess. *The Communications of the TeX Users Group 9.* (pp 284-287).

Buchgraber, G., Berndt, R., Havemann, S., Fellner, D.W. (2010). FO3D - Formatting Objects for PDF3D. In M.N. Zuffo et al. (Eds.) *Proceedings of the 15th International Conference on 3D Web Technology. July 24 - 25, 2010 Los Angeles, CA, USA* (pp. 63-72). New York: ACM.

ECMA International. (2007). *Standard ECME-363, Universal 3d File Format (4th edition)*. Retrieved April 4, 2011 from http://www.ecma-international.org/publications/standards/Ecma-363.htm.

Edwards, S.J. (1994). *Portable Game Notation Specification and Implementation Guide.* Retrieved April 4, 2011 from http://www.chessclub.com/help/PGN-spec.

Evans, L. & Wiswell T. (1953). *Championship chess and checkers for all*. New York: A.S. Barnes & Co.

FIDE – World Chess Federation. (2008). *Actual Handbook E.1.01B. Appendices C. Algebraic notation.* Retrieved April 4, 2011 from http://www.fide.com/component/handbook/?id=125&view=article.

ISO 19005-1:2005. (2005). *Document management – Electronic document file format for long-term preservation – Part 1: Use of PDF 1.4 (PDF/A-1)*. ISO, Geneva, Switzerland.

ISO 32000-1:2008. (2008). *Document management – Portable document format – Part 1: PDF 1.7*. ISO, Geneva, Switzerland.

Khronos Group. (2011). *WebGL Specification Version 1.0*. Retrieved April 4, 2011 from https://www.khronos.org/registry/webgl/specs/1.0/.

W3C. (2006). *Extensible Stylesheet Language (XSL) Version 1*. Retrieved April 4, 2011 from http://www.w3.org/TR/xsl11/.

W3C. (2009). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) specification*. Retrieved April 4, 2011 from http://www.w3.org/TR/CSS2/.

W3C. (2011). *HTML 5*. Retrieved April 4, 2011 from http://www.w3.org/TR/html5/.