Compiling 3D Models of European Heritage from User Domain XML

Brian Farrimond , Robina Hetherington
Liverpool Hope University College
{ farrimb@hope.ac.uk, hetherr@hope.ac.uk }

Abstract

Europe has a unique heritage and culture that is largely hidden from its young people. New information and communication technologies present an opportunity to involve the young people of Europe in discovering this heritage and presenting it to others in an exciting, dynamic way.

This paper describes early work on the INHERIT project which involves the development of a set of tools, data and structures to build and manage historical and three-dimensional models. This will enable school and college students to share in creating and exploring distributed simulations of dynamic aspects of history, geography, economics, politics and other subjects closely associated with European citizenship. With the tools developed it is envisaged that school pupils will be able to add, for example, landscape, buildings, avatars and a range of other objects, to a virtual world that models the heritage of Europe. The tools are based upon XML technologies to structure and distribute data and X3D or VRML to display that three-dimensional data.

Keywords--- **Information Visualization, Interactive 3D Graphics, X3D, XML, VRML, Cultural Heritage**.

1 Introduction

The time is now right for much wider applications of Web3D graphics. The growth of the use of broadband Internet connections and a significant rise in the number of relatively low-priced computers readily available, which can handle both the file size and rendering requirements of 3D models, mean that 3D worlds can be enjoyed by many.

The use of 3D is now standard in computer games and is now an expectation of the younger generation for computer graphics. This means that people, under the age of say 25, have very high levels of computer literacy and the ability to conceptualize in three-dimensions. Many games now incorporate the ability to create or add to three-dimensional environments. (Arendash(2004)). However few tools exist to allow school pupils to create models of their own environment.

The INHERIT Project seeks to develop a set of tools, data and structures to enable school and college students to share in creating and exploring distributed models of their cultural heritage. INHERIT will allow pupils (and teachers) to collaborate and communicate on-line while exploring and interacting with the INHERIT simulations. The 3D simulations will be built within the context of a collaborative environment enabled by the Internet. The tools are intended to be simple to use enabling students to generate rapidly 3D representations of specific historic events or environments. This paper describes early work in developing the data structures that will underlie these tools.

Churches were chosen as an initial subject of investigation for modelling. Every settlement across Europe has a church. It is usually the oldest building in the settlement and contains much of the local cultural heritage. The older, Gothic churches were also built according to quite strict rules. Whilst each individual church will have its own individual elements, it will be built up of a standard set of components, such as a nave, tower, transepts. These rules mean that the structure can be componentized and the individual characteristics can be stored as parameters. For instance a tower can be round or square, it can be topped with a steeple or roof, and it can be given dimensions.

The set of tools described are built using XML (eXtensible Markup Language) technologies. XML has been designed to meet the challenges of large-scale electronic publishing. XML's design goals, as set out by the W3C (2004) make it ideal for storing the components of historical structures and their parameters. The XML file is easy to create and can be processed for display over the Internet. The display of the object is with X3D (eXtensible 3D) or VRML (Virtual Reality Modelling Language). X3D, an application of XML, has recently received ISO International approval, is the new standard designed to enable real-time communication of 3D data,

Proceedings of the Ninth International Conference on Information Visualisation (IV'05)
1550-6037/05 $20.00 © 2005 IEEE

in particular over the Internet. (Trevett (2004)). However, as X3D is still relatively new and browser plugins are still in development and are not as widely used as VRML plugins, the XML data is also translated into VRML.

The objectives of this initial stage of the INHERIT project are:

- To develop a system to subdivide common structures such as churches into a set of components
- To develop a data structure to store the components and their associated values as parameters
- To develop a system to process this data into 3D formats, VRML or X3D.

This paper first outlines related work in the field of XML technologies and 3D objects and how 3D is being used to enhance and reinforce learning experiences. It then explains the rationale behind the XML technologies used in INHERIT and how they have been applied to the generation of models of churches. The data structures for these buildings are described. Finally conclusions are drawn as to future developments of the building tool.

2 Related Work

In the education field there a number of approaches to the application of XML technologies in the generation of three-dimensional representations of real life objects. Polys (2003) has applied XSLT (eXtensible Stylesheet Transformations) technologies to create multiple visualizations of the same data. He used data structured in CML (Chemical Markup Language) as the source of molecular information to generate the visualizations. He demonstrated that a highly structured format, CML, provides a format for the conversion and sharing of data files.

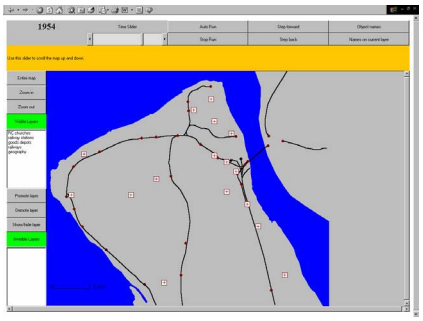

Figure 1 - 2D Time Map: Railways and Churches

Farrimond et al (2003) have demonstrated the use of XML based Time Maps to allow the presentation of a sequence of events in spatial context (Figure 1). This work enabled historical events, economic and political activities and other developments to be modelled and studied in a static and relatively non-interactive fashion across the World Wide Web. These 2-d simulations were developed and tested by a number of University students.

The Time Map work involved the use of historical data contained within XML with objects stored with time stamped attributes such as position, size, political control. These attributes were then displayed in the generated display to indicate different times or states of the event or object. For instance the various stages of construction of the railway shown in Figure 1 are displayed in different colours. Further Time Map work used XSL (Extensible Stylesheet Language) to generate VRML from the XML. (Farrimond et al. 2004) This work develops the use of XML to contain data but explores alternative technologies to process the XML.

Hetherington et al (2004) have demonstrated the use of XML and X3D technologies to integrate temporal data within X3D models and display buildings in various states or times accompanied by historical data. XML technologies were applied to structure the data or model file, in order that it could be filtered and displayed according to different criteria, in their case, time periods.

The 3D Virtual Buildings Project described by Bonnett (2003) enables students to generate representations of historic settlements using 3D modelling software; to explore how multiple methods of knowledge representation could be used to enhance student learning outcomes; and more specifically, to determine how 3D, and the process of 3D model construction, used in conjunction with text, could help students to realize a fundamental point, that historical representations are models, models that must be distinguished from the objects they purport to represent.

Wojciechowski et al (2004) have demonstrated a system to enable museums to build and manage virtual exhibitions of artifacts through X-VRML, a high level XML-based language. The X-VRML language applies the concept of parameterization of scenes to generate three-dimensional representations of objects within a virtual museum. The parameters can define elements of the scene such as for instance wall coverings. The ARCO system was developed to enable the exploration of museum artifacts by different audiences, in particular children, enabling museum staff to build interactive learning presentations.

Whilst the use of 3D to represent objects to students in an interactive manner to encourage them to engage and explore these objects in a meaningful way is important, the main objective of INHERIT is to develop a tool which will enable school pupils to create models of their own environment. Underpinning the INHERIT

Proceedings of the Ninth International Conference on Information Visualisation (IV'05)
1550-6037/05 $20.00 © 2005 IEEE

Project is a belief that modelling real world entities (such as individual buildings in the micro-scale or urban systems in the macro-scale) compels the modeller to look at the modelled entity with a different level of intensity.

Arendash (2004) applies a Game technology, Unreal Editor, to enable non specialist modelers to generate their own 3D environments. He demonstrates that anyone, as opposed to a specialist modeller using an expensive 3D development tool such as Maya or 3ds max, can create "*rich, compelling and very lively 3D experiences*". This paper explores the data structures necessary to treat a building, such as a church, as a collection or assembly of components. The next stage will be to apply a user interface that is intuitive and engaging for school pupils to use, learning from Game technology.

3 Inherit System Overview

The aim of component-based modelling is to provide descriptions of artifacts expressed in terms of their components that can be used to generate visualisations in 3D languages such as VRML and X3D. This is analogous to general programming principles in which coding is expressed in terms that are problem based and paradigm based. For example, Fortran scientific programs are expressed in terms scientists and engineers understand that are relevant to the problems they are trying to solve. Equally, programming languages in the object oriented paradigm enable developers to think and express themselves in terms of classes and objects. In both cases the nature of the underlying hardware is hidden and the developer does not need to understand it. Also, migration of a program to a new host is feasible if the appropriate compilers are available. This paper aims to demonstrate how the same principle can be applied to 3D modelling. The XML descriptions created by the user are expressed in terms the user understands and are related to the building they are trying to model. The XML descriptions are the translated into the target 3D language automatically by compilers. The user does not need to understand the 3D language. As new visualisation languages become available, new compilers can be developed that are used to translate the descriptions into the new language.

A benefit of this approach is that if the compiler is client side, the server need only serve the user's description to the client. The client compiles the visualisation and displays it. Generally, description files are much smaller than the equivalent VRML or X3D. This is equivalent to the X3D **<Sphere>** being able to specify the equivalent of an **<IndexedFaceSet>** with a large number of vertices. For instance the XML file for the simple church illustrated in Figure 2.is 11Kb in size, once compiled into a 3D model file it is 49 Kb and this is for a model with simple geometries; once applied to a structure with complex geometries, such as curves, much greater compression ratios can be achieved. The model shown in Figure 20 has an XML Description file size of 50 Kb and a model file of 388 Kb.

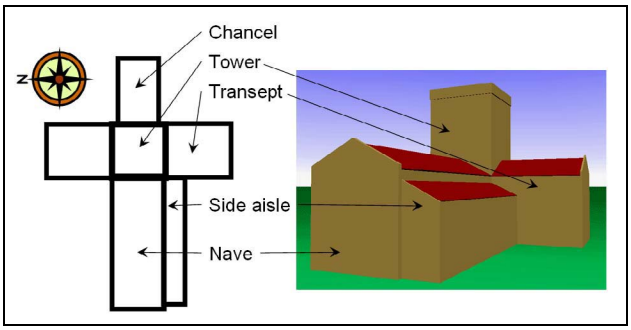

Figure 2 Plan and model illustrating the standard church components

Finally the description provides a natural storage mechanism for all kinds of information about the artifact including photographs, sounds, text and links to web resources. The compiler only uses the relevant part of the description but may be extended to make use of this other information, especially if it is attached to particular components.

Component-based modelling depends upon an analysis of real artifacts, identifying components of those artifacts that have a common underlying structure customised by parameters. Traditional 3D modelling will, for example, use a **<Box>** node customised by giving specific values for its length, breadth and height. European churches and cathedrals of the Romanesque and Gothic types form a suitable vehicle for such analysis. These buildings conform to a number of standard design features, for example all such churches have a nave, many have chancels, some have transepts and side aisles, towers and spires, see Figure 2. The orientation and positioning of these components are standard across Europe. The nave is aligned west-east, the chancel is at the east end of the nave with the same orientation, transepts are aligned north or south from the nave or chancel. Side aisles extend nave, chancel and transepts along their edges. Towers are placed at a number of places within this layout such as at the crossing of the nave and transepts, and on either side of the west end of the nave. Some churches do not follow this east west alignment but descriptions of these buildings refer to the alignment as being based on the "liturgical east".

Proceedings of the Ninth International Conference on Information Visualisation (IV'05)
1550-6037/05 $20.00 © 2005 IEEE

3.1 Analysing church structure as a set of rooms

In our work, analysis of the architectural style of traditional English churches reveals that their major components can be classified as variations of an underlying type which we call the **room**. The nave, chancel, transepts and towers can all be regarded as **room** type objects although in the XML, the user does not use the term room explicitly. Instead, the XML uses terms for the components that are meaningful in the context of churches: nave, chancel etc. The room comes into play when the XML is analysed and transformed by the compiler as described below. It is intended that the values of the XML attributes used to parameterise the components should be measurable from plans, photographs and on-site measurements using tape measures.

The XML description of a church building that identifies its major rooms is given in Figure 3

In the case of the **squaretower**, **chancel** and **nave**, **length** is west-east and **breadth** is south-north. In the case of the **northtransepts** and **southtransepts**, **length** is south-north and **breadth** is west-east. These can be assumed because of the standard layout of these churches as explained above.

```
<building name="test1.xml" units="feet">
   <church>
      <squaretower    length="18" breadth="22"
                      astdisp="-18" northdisp="0">
         :
         </squaretower>

      <nave           length="62" breadth="22"
                      eastdisp="0" northdisp="0">
         :
         </nave>

      <chancel        length="40" breadth="18"
                      eastdisp="62" northdisp="0">
         :
         </chancel>

      <southtransept length="40" breadth="18"
                      eastdisp="50" northdisp="-11">
         :
         </southtransept>

      <northtransept length="40" breadth="18"
                     eastdisp="20" northdisp="11">
         :
         </northtransept>

   </church>
</building>
```

Figure 3 The XML description of a church building that identifies its major rooms

Other types of **room** will be added as the project develops. These include porches, Lady Chapels and Galilee chapels as at Durham Cathedral and polygonal plan chancels such as at Gloucester and Norwich cathedrals.

The description of a **nave** as an example of a **room** in terms of its sub-components is given in Figure 4.

```
<nave length="62" breadth="22" eastdisp="0" northdisp="0">
   <roofsouth roofthickness="0.5" apexheight="48.4"
              baseheight="38.3" horizontalprojection="11"
              baselength="60" leftapexprojection="0"
              rightapexprojection="0" />

   <roofnorth roofthickness="0.5" apexheight="48.4"
              baseheight="38.3" horizontalprojection="11"
              baselength="60" leftapexprojection="0"
              rightapexprojection="0" />

   <wallwest basewallthickness="1" leftheight="40"
             rightheight="40"              apexheight="50.1"
type="gable">
   :
   </wallwest>

   <wallsouth basewallthickness="3" leftheight="38.3"
              rightheight="38.3" type="rectangular">
   :
   </wallsouth>

   <wallnorth basewallthickness="3" leftheight="38.3"
              rightheight="38.3" type="rectangular">
   </wallnorth>

   <sideaislenorth length="17" leftoffset="1" breadth="15"
                   roofthickness="0.5" isfullgable="false">
   :
   </sideaislenorth>

   <walleast basewallthickness="1" leftheight="40"
             rightheight="40"              apexheight="50.1"
type="gable">
   :
   </walleast>
</nave>
```

Figure 4 An example of a room in terms of its sub-components

3.2 Roofs

Roofs are located on each of the four edges of the **room**: north, south, east and west. The roofs rise from the room's wall to an apex. At present, roofs are base components with no further sub-division. Hence the element has attributes but no sub-elements. In general, the roof is a 2-D plane with a thickness specified by the **roofthickness** attribute. It is regarded as having its upper edge horizontal at a height above the ground given by **apexheight**. Its lower edge is also horizontal at a height given by **baseheight**. The other attributes provide the opportunity to have the upper edge longer or

Proceedings of the Ninth International Conference on Information Visualisation (IV'05)
1550-6037/05 $20.00 © 2005 IEEE

shorter than the lower edge to represent, for example, two roofs meeting at right angles along a common guttering as shown below in Figure 5

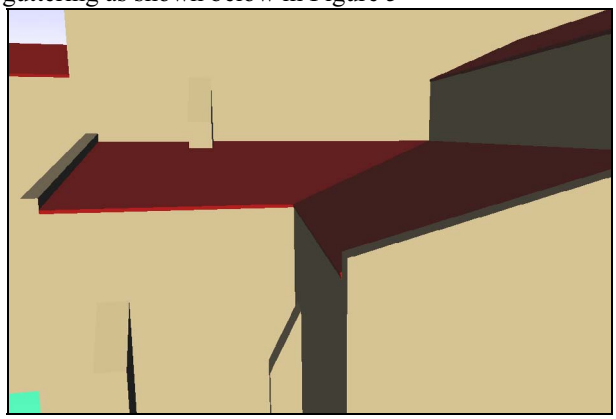

Figure 5 Showing a detail of a model with two roofs meeting at right angles

3.3 Walls

A **room** may have walls along each of its edges: north, south, east and west. The **wall** attributes identify the ground plan thickness through the **basewallthickness** attribute. They also provide for walls whose top edges slope rather than are horizontal. Two main types of wall are identified through the **type** attribute. These are **gable** and **rectangular**. The rectangular type may be rectilinear or trapezoidal as shown in Figure 6.

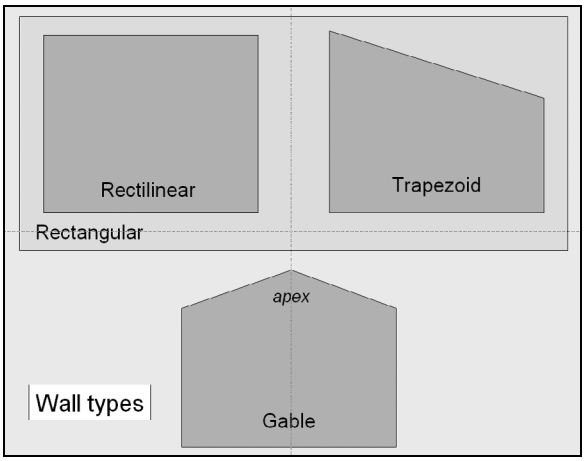

Figure 6 Wall types

A gable wall has an apex somewhere along its top edge as shown in Figure 6

Walls are more complex components than roofs and have a number of sub-components. These are currently: **wallsection**, **arcadesection** and **buttress**. An example is given in Figure 7. The wall is considered as a number of wall sections and arcade sections stacked on top of each other. The **id** attribute determines the stacking order.

```
<wallsouth basewallthickness="3" leftheight="38.3"
          rightheight="38.3" type="rectangular">
   <arcadesection id="0">
      <wallsection id="0" type="rectangular"
                  wallthickness="3" leftheight="33"
                  rightheight="33" >
      :
      </wallsection>

      <column id="0" leftoffset="3.5" type="cylinder"
                   h="10" r="1.5" />

      <column id="1" leftoffset="21.5" type="cylinder"
                   h="10" r="1.5" />
   </arcadesection>

   <wallsection id="1" type="rectangular"
               wallthickness="2" leftheight="5.3"
               rightheight="5.3">
   :
   </wallsection>

   <buttress id="0" centreoffset="1" anglefromnormal="45">
   :
   </buttress>

   <buttress id="1" centreoffset="21">
   :
   </buttress>
</wallsouth>
```

Figure 7 XML fragment containing wall sub-components

wallsection: each wall section has its own thickness. Optional wall section attributes enable sections to project over adjacent wall sections. Each wall section may contain arches and windows - see below.

arcadesections: an arcade section consists of a wall section plus any number of columns. The **leftoffset** attribute for a column determines the position of the column relative to the left hand edge of the wall section as seen when facing the wall length.

buttresses: a buttress is placed along the wall at a distance from the left edge of the wall determined by the **centreoffset** attribute. The optional **anglefromnormal** attribute determines the angle in degrees from the normal to the owning wall. Most buttresses have an **anglefromnormal** of 0. Buttresses at corners are often at 45 degrees to the normal as shown in Figure 8.

The buttress is built up from sections just as the wall is built up from sections as shown in Figure 10. Each section has parameterised knee, hip heights and depths. The plan of the sections is currently rectangular.

Proceedings of the Ninth International Conference on Information Visualisation (IV'05)
1550-6037/05 $20.00 © 2005 IEEE

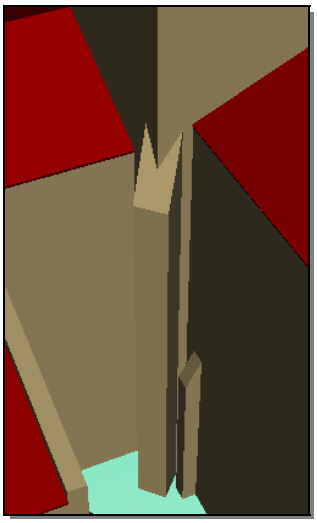

Figure 8 Detail of a buttress at a corner

A complex buttress can be created by stacking sections and by choice of knee and hip depth and heights. Figure 10 illustrates how two sections are stacked to form a buttress. Figure 9 shows the corresponding XML.

```
<buttress id="0" centreoffset="2">
   <buttresssection id="0" width="2"
               heightback="10" depthhip="1.7"
               heighthip="10" depthknee="2.5"
               heightknee="9" depthbase="3.0" />
   <buttresssection id="1" width="2"
               heightback="15" depthhip="1.7"
               heighthip="14" depthknee="1.7"
               heightknee="14"    depthbase="1.7"
/>
</buttress>
```

Figure 9 The XML description of a two section buttress

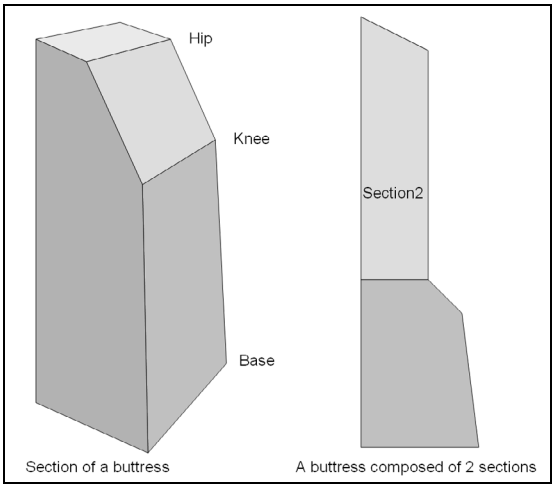

Figure 10 Building a buttress

3.4 Arches

Arches are specified as part of a wall section. The arch attributes identify the distance of the arch from the left end of the wall section, the height of the base of each side of the arch (which may rest on the top of columns) and measurements that enable the profile of the arch to be determined. These measurements for the left side of the arch are illustrated in Figure 11.

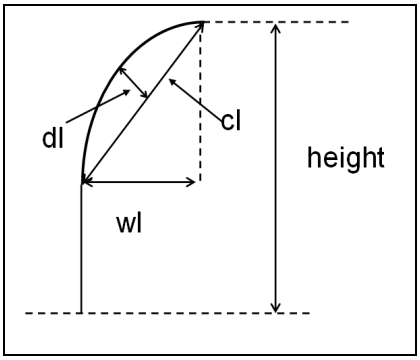

Figure 11 Measurements for the left side of an arch

Measurements for the right side of the arch are also recorded in attributes. The profile of the arch cross section is specified by using indents and offsets as shown in Figure 12.

```
<archid = "0" leftoffset = "6" leftfootheight = "0"
              rightfootheight = "0"
              cl = "15" wl = "5" dl = "1"
              height = "28"
              cr = "15" wr = "5" dr = "1"
              insetfront =    "0.2,    0.5 ,  0.8,  1.2"
              indentfront =   "0.2,    0.3,   0.4,  0.8 "
              insetback =     "0.2,    0.5,   0.8,  1.2"
              indentback =    "0.2,    0.3,   0.4,  0.8"/>
```

Figure 12 XML for an arch

The corresponding cross section is shown in Figure 13.

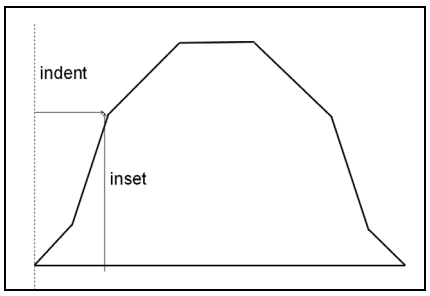

Figure 13 Cross section of an arch

The cross section of the soffit is given by inset and indent pairs, **insetfront** and **insetback**, where inset is the distance from the outer arch towards the centre of curvature while indent is the horizontal distance into the wall from its outer surface. The front and back of the arch are specified separately. The effect of the description is shown in Figure 14.

While this meets the needs of many architectural styles, further work is needed to describe successfully more elaborate cross sections that make use of rounded moulding.

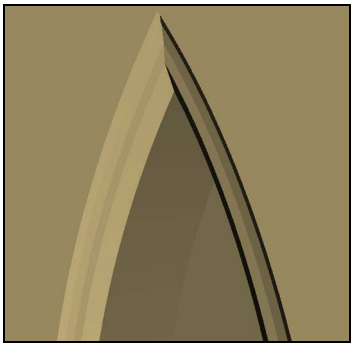

Figure 14 Model of an arch

Thus in the photo in Figure 15 showing an arch in the transept of Wenlock Priory, the upper section can be described with insets and indents. However, the lower section with its rounded moulding requires a different approach that has yet to be implemented.

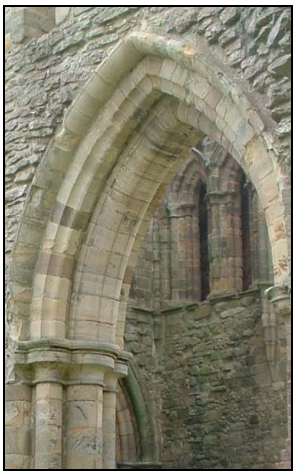

Figure 15 Photograph of an arch in the transept of Wenlock Priory

3.5 Windows

A window is seen as having an arch as a sub-component. Figure 16 gives the XML for a window. Circular or rose windows have yet to be implemented.

```
<window id="0" sillheight="1">
   <arch id="0" leftoffset="41" leftfootheight="0"
               rightfootheight="0"
               cl="2" wl="1" dl="0.2" height="4.5"
               cr="2" wr="1" dr="0.2" />
</window>
```

Figure 16 XML code for a window

3.6 Side aisles

A side aisle can be attached to any wall of a **squaretower**, **nave**, **chancel** or **transept**. The offset from the left end of the wall is specified by an attribute. Sub-components of a **sideaisle** can include walls and roofs.

3.7 Spires

A square tower may be surmounted by a spire. Figure 17 gives the specification of an octagonal corned spire.

```
<spire id="0" type="octagonalcornered" baseheight="92.7"
             length="18" breadth="18" height="50"
             wallthickness="2" cornerslopeheight="10">
</spire>
```

Figure 17 The specification of an octagonal corned spire

The **cornerslopeheight** gives the distance up the slope of the apex of the corner triangle as illustrated in Figure 18.

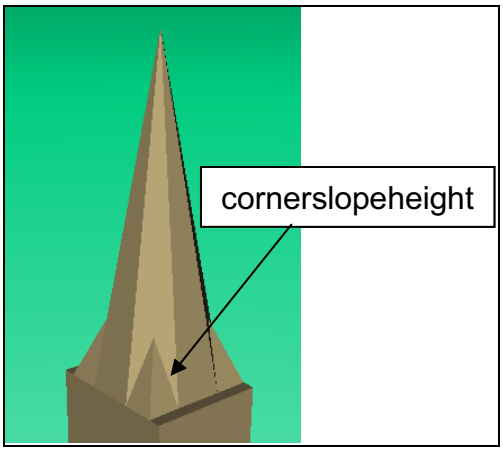

Figure 18 Detail of the junction of an octagonal spire and a square tower

This section has covered the data structures used to build the models, examples of which can be seen in Figure 19 and Figure 20. As can be seen some quite complex and varied forms can be created using this technique.

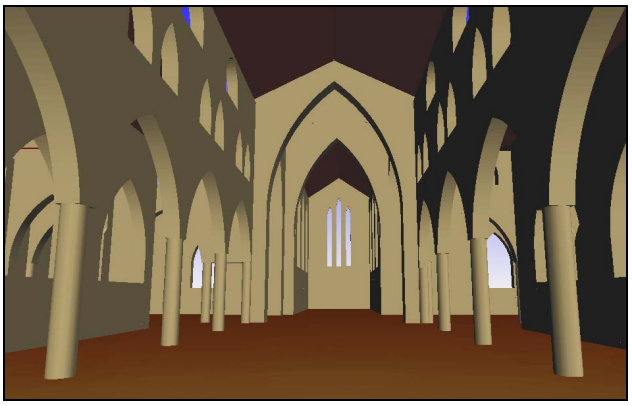

Figure 19 Interior of a model of typical gothic church architecture

Figure 20 Exterior of a model of typical gothic church architecture

3.8 The development environment

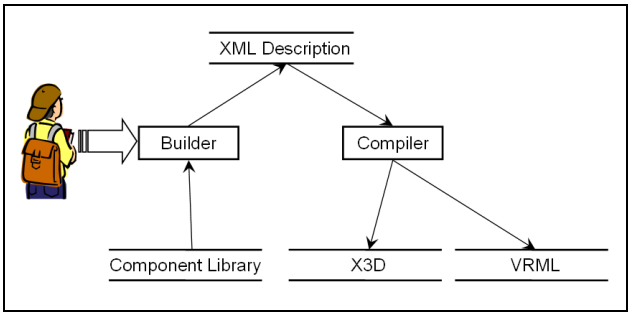

Figure 21 Diagram of the compiler process

The strategy of the project is to enable a user to create a description graphically from components provided in a Component Library. The approach will be similar to most graphical editing tools except that the components will not be abstract entities such as lines, circles, spheres and boxes but will be representations of real world objects. The description will be stored as an XML Description. This description can then be compiled to generate X3D or VRML as illustrated in Figure 21.

So far the work has concentrated on the compiler section of the system to prove the concept. The XML Description is edited using Notepad++ and CookTop.

3.9 The compiler

The compiler is written in C, making use of the Xerces C++ validating XML Parser (Apache XML Project). Earlier work by Polys (2003), Farrimond et al (2004) and Hetherington et al (2004) all used XSL technologies to transform or parse XML data. However, after initial consideration to use XSL to transform the XML Description into X3D or VRML, it soon became apparent that the mathematical processing required to generate the 3D models was too complex to achieve with XSL. C proved to be a more satisfactory choice while proving the concept.

Having read and parsed the XML Description, the compiler walks through the resulting DOM data structure generating its own internal representation based upon a C++ ROOM data structure shown in Figure 22. The compiler has analysed the XML description into types of room.

```
typedef struct {
        double eastdisp;
        double northdisp;
        double length;
        double breadth;
        bool hasspire;
        SPIRE spire;
        bool hasroof[4];
        ROOF roof[4];
        bool haswall[4];
        WALL wall[4];
        bool hassideaisle[4];
        SIDEAISLE sideaisle[4];
} ROOM;
```

Figure 22 Data structure for ROOM used in internal representation

The SPIRE, ROOF, WALL and SIDEAISLE data types have equivalent structures.

The compiler then uses the internal representation to generate the VRML or X3D output. The structured nature of the XML Description lends itself to mapping onto the structured nature of VRML and X3D. The **`room`** becomes a **`<Transform>`** node and the room sub-components such as walls and roofs are children of the room node. The names of the XML description tags such as **`wallsouth`** in conjunction with the values given in the attributes are used to identify rotations and translations. The wall sections and arcade sections become children of the wall nodes.

Proceedings of the Ninth International Conference on Information Visualisation (IV'05)
1550-6037/05 $20.00 © 2005 IEEE

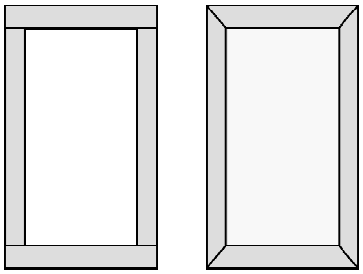

Figure 23 Wall junctions

Currently, the compiler builds the walls of a room in the fashion shown in plan form on the left of Figure 23. This follows the style in many churches but further development would enable the walls to be mitred at the corners as shown on the right of Figure 23.

The wall sections are drawn as **<IndexedFaceSet>** nodes with vertices determined by the arch information. Trigonometry uses the parameters for the arches to generate the vertices. The resulting shape is invariably concave and future work would examine how this may be refined to improve viewer performance. The roofs and spires are also built from **<IndexedFaceSet>** nodes**.**

The use of **<IndexedFaceSet>** is universal throughout the compiled VRML or X3D. It is intended that the Builder tool will provide facilities whereby the user can associate parts of photographs with parts of the walls etc and have the Builder tool build up composite images that would be used by the compiler to texture the model.

4 Conclusions

This paper covers the initial stage of the INHERIT project and much work remains to create tools to allow school pupils to be able to add a range of objects, to a virtual world that models the heritage of Europe

This paper explored how common structures such as churches can be divided into a set of components. The data structures to store the components and their associated values as parameters in XML are then outlined and finally a system to process this data into 3D formats VRML or X3D has been described.

There are a number of areas for future development including working with the level of detail of the models generated, and by default the choices offered with the XML Description. The possibilities for enhancing the richness of the model include the use of photographic textures. It is felt that ease of use for the non-expert will be a key aspect of the Builder tool. Other developments would provide facilities for modelling the tracery and stained glass of the windows, the modelling of the vaulting, catering for steps between different levels in some churches and handling the embellishment of walls and arches.

Further work would also be to compare a compiler written in Java with the C compiler used in this study. Java is the language of choice for Web projects and a comparison of the relative strengths of the two programming approaches should be undertaken.

The lessons learned during this early work will inform into the detailed development work on the more general modelling tools and data structures that will underpin the INHERIT project.

5 References

APACHE XML Project, http://xml.apache.org/xerces-c/, accessed 5th November 2004.

ARENDASH, D., 2004, The Unreal Editor as a Web 3D Authoring Environment, *Proceedings of the Ninth International Conference on 3D Web Technology*, ACM Press, New York, NY, USA, ACM, 119 – 126.

BONNETT J. (2003) Following in Rabelais' Footsteps: *Immersive History and the 3D Virtual Buildings Project*, Journal of the Association for History and Computing Vol VI, Number 2. Matsushita Center for Electronic Learning, Pacific University, Forest Grove, Oregon, U.S.A.

FARRIMOND, B. and HETHERINGTON, R., 2004, Using 3D to Visualise Temporal Data, IV04, *Proceedings of the Eight International Conference on Information Visualisation*, IEEE Computer Society, Los Alamitos, California, 108- 117.

FARRIMOND, B., PARKINSON, L., and POGSON F., 2003 Modelling history with XML, *DRH 2001 and 2002*, OHC, London. J. Anderson, A. Dunning and M. Fraser Ed, 89 to 111.

HETHERINGTON, R. and SCOTT, J. P., (2004) Adding a Fourth Dimension to Three Dimensional Virtual Spaces, *Proceedings of the Ninth International Conference on 3D Web Technology*, ACM Press, New York, NY, USA, 163-172.

POLYS, N. F. 2003, Stylesheet Transformations for Interactive Visualisation: Towards a Web3D Chemistry Curricula, Proceedings of the Eighth International Conference on Web3D Technology, ACM Siggraph, 85-90.

TREVETT, N. WEB3D CONSORTIUM, 2004, Web3D Consortium - Press Releases, http://www.web3d.org/news/releases/archives/000092.html, accessed 5th November 2004

W3C, 2004, Extensible Markup Language (XML) 1.0 (Third Edition) http://www.w3.org/TR/REC-xml/, accessed 5th November 2004

WOJCIECHOWSKI, R., WALCZAK, K., WHITE, M. and CELLARY, W., 2004, Building Virtual and Augmented Reality Museum Exhibitions, *Proceedings of the Ninth International Conference on 3D Web Technology*, ACM Press, New York, NY, USA, 135 - 144

WEB3D CONSORTIUM, X3D OVERVIEW, http://www.web3d.org/x3d/overview.html, accessed 5th November 2004