Developing a Parametric Approach for 3D Modelling Software

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

Abstract

The creation of 3D models is generally considered by newcomers to be a difficult activity requiring a number of skills and considerable practice.

This paper describes work in the INHERIT project which aims to address these issues by providing a 3D modelling tool set which is easy to use, requiring few skills and little practice. This is achieved by the development of software tools which are customised to build particular types of model.

The key aspect of these tools is the treatment of the underlying data of the 3D model as a tree structure of nodes which consist of parameterised representations of the components of the object being modelled. The tools then automatically generate the graphics primitives that enable the visualisation and interaction with the object. This paper describes the implementation of the first tool created following this principle which enables schoolchildren to model church structures.

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

1 Introduction

The INHERIT project seeks to develop a set of tools to enable school children to share in creating and exploring distributed models of their cultural heritage. INHERIT will allow pupils (and teachers) to collaborate and communicate on-line by exploring and interacting with the INHERIT simulations. The 3D simulations will be built within the context of a collaborative environment enabled by the Internet. (Farrimond et al (2005))

The tools are intended to be simple to use, enabling children to generate rapidly 3D representations of specific historic events or environments.

In order to achieve this, the tools would need to meet these goals must be:

- simple to learn
- quick to model with
- easy to add information about the model
- inexpensive

These goals are imposed by the realities of life in schools which have limited resources in terms of time within the curriculum available for children to learn how to use tools and apply them. The design goal of *simple to learn* was further refined to enabling a child to learn how to use the tool in 15 minutes and create a useful model within another 30 minutes. Cost is another significant factor affecting the uptake by schools.

Standard 3D modelling software, because of its expense and long and steep learning curve, is not suitable to meet the INHERIT goals. It was decided to approach the problem from a different perspective. High level language and object oriented software development paradigms provide alternative approaches. In both cases, the method is to hide away the mechanics of the machine on which the software is to run and instead to have the developer focus only on problem domain concepts. A programmer in Fortran or Java does not need to be aware of the architecture of the computer on which their programs run. Instead the programmer creates and manipulates variables or objects which represent real world characteristics and entities. This paper outlines how a 3D modelling tool has been developed using these concepts. Using the software the modeller is able to create and manipulate representations of real world entities and not be concerned with the graphical primitives such as vertices, lines and polygons with which the entities might be visualised.

The modeller needs direct control over graphical primitives in specific circumstances. For example, the control of polygon counts is an essential aspect of successful modelling since using too many polygons will result in a model that is too slow to render in real time on a typical machine. However, once the modeller has specified in their creation and manipulation of the modelled object's components what they want to happen

Proceedings of the Information Visualization (IV'06)
0-7695-2602-0/06 $20.00 © 2006 IEEE

(information level design), the modelling tool is able to generate automatically an efficient collection of primitives to achieve its visualisation (implementation level design).

This work builds upon early work by Farrimond et al (2005) where the data structures were developed to record the parameters of a church building. The hierarchical structure and flexibility of XML provided the natural expression of the data structures.

Churches were chosen as an initial subject of investigation for modeling. It is envisaged that many more tools will be developed to model such diverse objects as mosques, solar systems, ships hulls, etc. Churches were chosen as 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 strict rules. Whilst each church will have its own individual elements, it will be built up of a standard set of components, such as a nave, tower, transepts (Fletcher (1954)). 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.

This paper first outlines related work in how 3D is being used to enhance and reinforce learning experiences and the tools available to support it. It then explains the rationale behind the design of the INHERIT tools and describes the implementation. Finally conclusions are drawn as to future developments of the INHERIT tools.

2 Related Work

Children, because of the prevalence and widespread use of video games, have excellent abilities to visualise in three dimensions. The use of 3D to represent objects to students in an interactive manner encourages them to engage and explore these objects in a meaningful way. Aguilera et al (2003) states that 3D in gaming, encourages observation in children, especially visual and spatial discrimination.

The ultimate interaction must, however, be to actually create a 3D model. The importance of modeling environments by students is discussed by Bonnett (2003) in the 3D Virtual Buildings Project. The process of 3D model construction, used in conjunction with text, is shown to demonstrate that multiple methods of knowledge representation enhance student learning outcomes The INHERIT software toolset enables school pupils to create models of their own environment. Underpinning the INHERIT 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.

However, whilst children have strong visualization skills, commercial 3D modelling tools are generally too difficult for them to use successfully. The majority of the facilities they provide will not be used, or even understood, by most school children.

Arendash (2004) demonstrated that anyone, as opposed to a specialist modeler using an expensive 3D development tool such as Maya or 3ds max, can create "rich, compelling and very lively 3D experiences". He applied the game technology, Unreal Editor, to enable non specialist modelers to generate their own 3D environments.

Other examples of 3D modelling tools for use by non-experts are commercial software such as Microsoft's Train Simulator. The train simulator provides a toolkit which enables enthusiasts to customize and extend the product by creating new trains, routes, scenery, and challenges. According to Henkel (2003) users often prefer carrying out development work to using the simulation; they describe the process as mass customization of the product.

Hetherington et al (2006) outline the need for the augmentation of sensory rich virtual environments with abstract information. In the context of modeling cultural heritage the information can contain important data such as dates, historical background, material and even anecdotal comments.

This paper demonstrates how the data structures discussed by Farrimond et al (2005) can be used to create easy to use 3D modelling software which can then be used to produce information rich virtual environments. It outlines the components of the church structure, the development of the software, in particular the development of the User Interface. It describes decisions made regarding working with parameters and providing different views and additional textural to the model. Finally it explains the method for publishing the models.

3 Description of Church Builder

Component-based modelling relies upon the artifacts that have a common underlying hierarchy. European churches and cathedrals of the Romanesque and Gothic period all contain a set of particular features, for example they all have a nave, many have chancels, some have transepts and side aisles, towers and spires. These are all arranged in a typical manner. (Farrimond et al (2005))

3.1 Church components

The hierarchy of components modelled in Church Builder is as follows:

church
- nave
- crossing tower
- chancel
  - wall
    - tower
    - transept
    - porch
    - buttress
      - buttress section
    - wall section
      - arch
      - column
      - picture

Note that towers, transepts and porches themselves have wall components. Development consisted of finding the best way of enabling the modeller to assemble a church from these components.

3.2 The development environment

Church Builder was developed using Microsoft Foundation Classes within Visual Studio. The programming language was C++. The graphics are implemented in OpenGL. Data is stored as an XML format, as described in their earlier work by Farrimond et al (2005). The software can also used to compile an X3D or VRML file for sharing of the model over the Internet. The compiler process is shown in Figure 1.

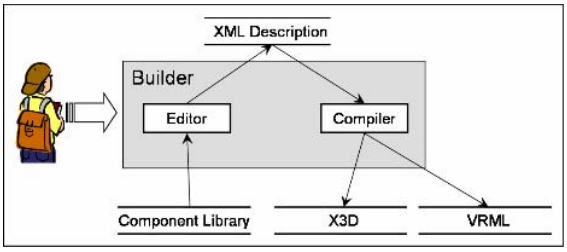

Figure 1 Diagram of the compiler process

Church Builder has gone through a number of transformations leading towards the current product. In its first format, the modeller used a Wizard to build the church. After testing in schools, the Wizard approach was abandoned and replaced by a single dialog box incorporating a tree control.

3.3 First attempt - using a Wizard

The concept of using a Wizard is a well established and successful metaphor in computing and enables the user to undertake complex tasks in a series of well defined steps with the possibility of retracing and amending earlier steps. This seemed appropriate for constructing a building and consequently, in Church Builder, the logical steps were defined to be:

1) main structure (optional nave, optional crossing, optional chancel)
2) transepts
3) towers
4) sideaisles
5) porches
6) wall details: - arches, buttresses, windows, doors.

Figure 2 shows the Wizard's first page in which the modeller can add a nave, crossing tower and chancel which together comprise the spine of the church. The modeller can then set the length, height, breadth and wall thickness parameters of those components.

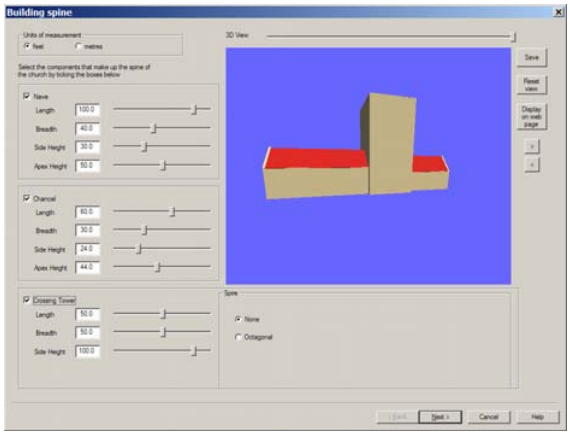

Figure 2: Builder Wizard first page

The right hand side mainly consists of the 3D view which can be manipulated by dragging the mouse to rotate and zoom in and out of the building. The left hand side consists mainly of edit box / slider pairs which are used to set and change the component parameters. In these cases: length, breadth, side height and apex height. Check boxes are used to add or remove a component from the church.

Figure 3 shows the last page of the Wizard in which the modeller can add arches, columns, windows and buttresses to walls. The design thinking followed the top-down design methodology of constructing the main parts of the building first then increasingly refining the building by first adding components such as transepts,

towers, sideaisles and porches to these main components then refining the individual walls.

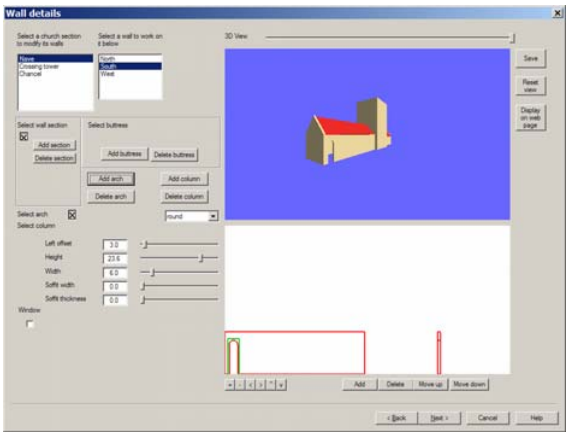

Figure 3: Builder Wizard last page

Figure 3 shows the 3D view of the building on the upper right and, at bottom right, a 2D view of the selected wall's elevation and cross section.

3.4 Critical appraisal of the Wizard approach

In the first trial at a primary school, the children were disappointed not to be able to add doors and windows as soon as they built the nave! This led to a re-evaluation of the interface.

The design had been based on what had appeared to be a systematic approach, which was not how the children wanted to model the churches. It is easiest to learn how to use a tool if it enables you to use it in a way that seems natural to you.

Feedback from demonstrations to teacher trainers indicated that the user interface on the individual Wizard pages was too complex. In Figure 2, for example, the parameters of three components (nave, crossing tower and chancel) are all visible at the same time.

In addition, selection required the manipulation of several list boxes. For example, Figure 3 shows a collection of list boxes and check boxes that are needed to select an arch belonging to a particular wall section on a particular wall on a particular part of the building. Selection of individual components is achieved on all Wizard pages by selection from one or more list boxes. This was not satisfactory.

3.5 Second attempt - single dialog

In response to this critical feedback, the interface was transformed into a new format consisting of a single dialog box. This is illustrated in Figure 4. In this interface the 2D window is always present underneath the 3D window on the right hand side of the dialog box. A tree control is employed at the upper left of the interface to help provide navigation for selection.

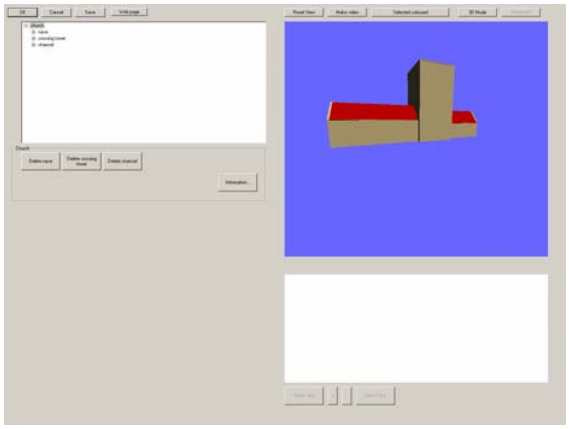

Figure 4: Single dialog interface

When a component such as the nave is selected, only that component's parameters are displayed for setting and amending. This is shown in Figure 5. This contrasts with the situation in Figure 2.

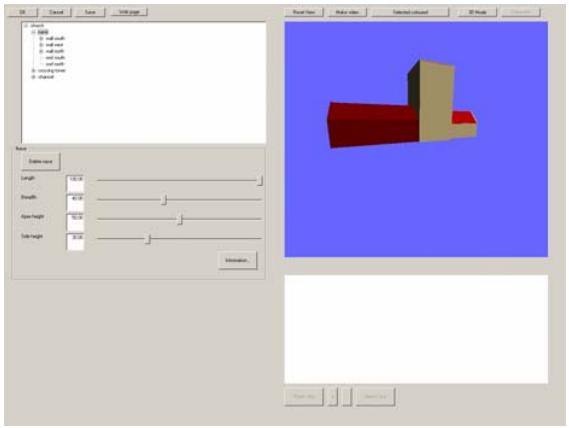

Figure 5: Interface with Nave selected

This feature of only showing the selected component's parameters applies to every component. In this way the interface clutter is systematically reduced. The feature reinforces the view that modelling consists of customising the parameters of the object's components.

The replacement of multiple list boxes by a single tree control also simplifies selection. It has the added advantage of reinforcing visually the learning point that a complex object can be regarded as an assembly of simpler components.

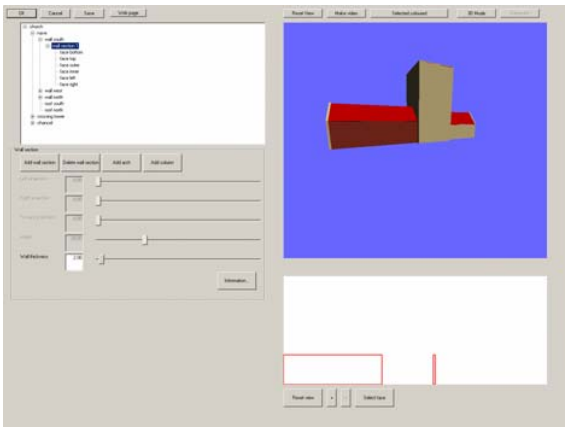

Figure 6: Interface with wall section selected

In addition to being able to access the parameters of a selected component, the interface displays buttons enabling the user to perform relevant actions within the context of the selection. For example, in Figure 5, a button can be seen that enables the user to delete the nave. In Figure 6 a wall section is selected in which context buttons are provided to add another wall section, delete the wall section, add an arch or add a column. Thus interface clutter is again reduced by only displaying buttons relevant to the selected item.

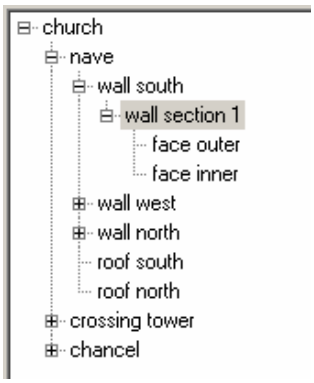

Figure 7: A selected wall section in the tree control

It was soon recognised that selection via the tree control alone was insufficient. A building of even quite moderate complexity soon accumulates many components and searching for them in the tree control was laborious. It was far simpler to select most components by clicking in the 3D. Selection in the 3D view automatically selects the corresponding node in the tree control. This is particularly useful for localising selection in the tree control. For example, selecting an arch is not yet implemented in the 3D view. However, selecting its wall section in the 3D view opens the corresponding node in the tree control revealing the arch node ready for selection as illustrated in Figure 7.

Selection of components in the 3D view proved difficult due to the hierarchical nature of the components. At present the convention used in the tool is that a left click selects the wall section, a right click selects the complete wall whereas to select the whole nave, for example, a right click on its roof is required. However, this is not intuitive and further research is needed into the possibility of a more appropriate solution. Use of the tree display has been found to be useful here in supplementing clicks on the 3D view.

One of the requirements of the INHERIT project is that it addresses special needs issues. The provision of alternative methods of selecting a component contributes to meeting this requirement.

3.6 Applying constraints to the setting of parameters

The parameter-based approach to modelling enables the tool to apply constraints because it knows the semantics of the components being added. This contrasts with generic 3D modelling tools which do not know the meaning of the shapes created. As an example of this, when editing an arch within a wall section in Church Builder, changing the value of the width parameter affects the lower limit possible for the height of the arch. For example, a round, Norman arch must be at least half as high as its width otherwise the geometry does not work. Church Builder can impose such constraints by changing the lower and upper bounds on the slider controls for the parameters.

3.7 Adding information about a component

The modeller can add information about the selected component using a button labeled *Information...*It can be seen in Figure 6, to the lower right of the parameter edit boxes and slider controls. This information could be of any textual type and might include a textual description or a URL linking the component to any kind of resource, for example photographs, videos, aural recordings, web sites. The saved file generated by the tool records this information and hence makes it accessible to a wide range of tools. This illustrates how the XML storage format providing semantic dimension to the model, transcends a purely graphical file format produced by generic 3D modelling tools by storing non-graphical information.

3.8 Alternative views

One of the issues with using commercial software packages is that they were generally designed with professional modellers in mind who may be modelling any kind of object. An advantage of creating specific builder tools, customised to creating a specific type of object, is that the number of facilities provided for the modeller can be drastically reduced. It also enables the tool developer to include views and tours appropriate to the object being modelled. In the case of Church Builder, a standard set of viewpoints is built in according to the components the modeller adds to his or her model. These are created without the need for the user to create viewpoints manually. Also, in the case of Church Builder a "Doom" type navigation has been added as an alternative to the standard rotating and zooming editing mode. Churches are quite natural places to tour on foot.

3.9 Alternative outputs

Alternative means of viewing the model have been developed, which will enable the INHERIT objective of using the Internet to distribute the models. The model can be published into a web page embedding a VRML file and is available at any stage of development of the model. Similarly, AVI movie clips can be generated by Church Builder.

4 Conclusions

This paper describes the first of a series of tools that will be used within the INHERIT project. The goals to be met are identified as being:

- simple to learn
- quick to model with
- easy to add information about the model
- inexpensive

Trials in schools have demonstrated that children upwards of 8 years old can learn how to use the tool within 15 minutes and can produce quite creditable buildings within 30 more minutes. This means that children can focus on the learning that the tools are meant to support rather then spend most of their time fighting the tools.

Because of the low demands made by the tool, it has been found to run successfully on older PCs in English primary schools even with Windows 98 installed and with screen resolutions as low as 800x600.

In the course of the development it has become clear that certain principles apply to tools of this type regardless of the kind of object they are designed to build. Future work will aim at automating the process of tool creation and hence reduction of costs and making more realistic the generation of a whole range of cheap builder tools that can run on low specification PCs.

4.1 Future work

The church builder tool is still under development, in addition to ongoing refinement and testing, future work will include:

- The addition of the ability to model buildings with a temporal dimension as discussed by Hetherington et al (2004) and Farrimond et al (2002). Buildings such as churches have been continually be altered with elements being added and removed at different times. The tool could be extended to allow this.
- The exploration of storage and retrieval of the completed models on a server with solutions such as using an API (Application Programming Interface) to Google Earth.

5 References

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., 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.

FARRIMOND, B. and HETHERINGTON, R., 2005, Compiling 3D Models of European Heritage from User Domain XML, IV05, *Proceedings of the Ninth International Conference on Information Visualisation*, IEEE Computer Society, Los Alamitos, California, USA

FLETCHER, B., 1954, A History of Architecture by the Comparive Method, 16th Edition, London, 333-334.

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.

HETHERINGTON, R., FARRIMOND, B. and PRESLAND, S., (2006) Information Rich Temporal Virtual Models Using X3D, accepted for publication in *Computers & Graphics*, vol.30 no.2

DE AGUILERA, M., and MÉNDIZ, A., (2003)Video Games and Education (Education in the Face of a .Parallel School.) *ACM Computers in Entertainment*, Vol. 1, No. 1, October

HENKEL, J., THIES, S. and JOACHIM, S., (2003) Customization and Innovation: User Innovation Toolkits for Simulator Software." *Proceedings of the 2003 Congress on Mass Customization and Personalization (MCPC 2003),* Munich.