Kiddy CAD: A Child Friendly Parametric Approach for 3D Modeling Software

Brian Farrimond[1], Janette Bonar-Law[2] and Robina Hetherington[3]

[1]Liverpool Hope University, farrimb@hope.ac.uk
[2]Liverpool Hope University, bonarj@hope.ac.uk
[3]Liverpool Hope University, hetherr@hope.ac.uk

ABSTRACT

3D models are increasingly used to visualize objects for presentation in schools. Children are sophisticated consumers of 3D worlds and engaging creatively with ICT in the classroom is recognized as beneficial to developing multiple intelligences in children. Historically, the creation of 3D models by children has, however, been considered too difficult a process to master. This paper describes the development of a CAD tool, Church Builder, designed for use by children, which is fast, fun and easy to learn and has minimal system requirements. The tool treats the underlying data of the 3D models as a tree structure of nodes, consisting of parameterized representations of the components of the objects being modeled. The tool automatically generates graphics primitives that enable the visualization of and interaction with the object, allowing children to rapidly build sophisticated model church structures.

Keywords: CAD applications, 3D graphics, VRML, XML, education.

1. INTRODUCTION

Children are sophisticated, 'digital native' [14] consumers of virtual reality and 3D gaming environments. Despite popular media speculation to the contrary, research suggests that these environments and their associated technologies may prove to be positive resources for learning [4]. The harnessing of ICT in the classroom to express and create ideas is recognized as beneficial in developing multiple intelligences in learners [13], [10] and CAD, VRML and 3D Graphics applications, in particular, offer the opportunity to engage logical-mathematical, visual-spatial, kinesthetic and naturalist intelligences in a combination, which could be viewed as almost unique. Why, then, are teachers and children not harnessing the benefits of modeling in 3D in the classroom?

The answer is relatively straightforward. In the UK Primary education sector (age range 5 – 11 years) ICT coordinators, responsible for determining what software is purchased and how ICT is incorporated across the curriculum, may not have begun their careers as IT or computing specialists. These 'digital immigrant' [14] gatekeepers to technology cannot be relied upon to have gained experience of industry standard modeling software and its conventions and are, consequently, unlikely to feel comfortable and confident with its use. Schools have limited time resources within the curriculum for both children and teachers to learn new tools, together with limited financial resources for purchasing new software. If teachers are not confident that they can easily use software and quickly develop materials to incorporate it into the curriculum then the software will not find its way into the classroom. This position improves in the Secondary education sector (age range 11- 18), where ICT is more likely to be taught by specialists. However, even these 'specialists' retain their 'digital immigrant' status, simply because they often pre-date the technology present in their classrooms. In addition to this, financial and time constraints persist, as does the historical belief that modeling in 3D is a process too complex for children to master. Under these circumstances, it is clear why the use of 3D modeling tools in a classroom setting is rare.

Previous work [5-6] has set out to develop a CAD tool set, Church Builder, to address these issues; a tool set which is simple to learn, which quickly builds easily manipulated models and is inexpensive and easy to deploy. This paper sets out how Church Builder was developed and subsequently altered to improve functionality and perceived application usability for digital immigrant gatekeepers and digital native learners.

2. RELATED WORK

Church Builder is not unique in attempting to make sophisticated modeling tools available to a broader range of users, and Church Builder, itself, evolved as part of a larger project, INHERIT [6] which, in turn, was prompted by a global trend towards digitizing cultural heritage

2.1 Other Tool Sets with Similar Aims to Church Builder

There is a long-standing tradition, fuelled by gaming advances, of trying to make 3D modeling tools more accessible to gamers as well as developers. Epic Games have included 'UnrealEd' [17], an IDE to create new game spaces, with its game 'Unreal' since 1997 and it is also included with 'Wheel of Time' and 'Rune'. Caligari produce a range of gamer/developer-targeted products including 'Truespace' [16] and 'Gamespace' [9]. 'Truespace' is a generic 3D authoring tool incorporating direct 3D manipulation controls, context based editing tools and a 'Magic Ring widget' for drawing 2D and 3D primitives. 'Gamespace' operates on a similar premise, but in this instance, models for export specifically to gaming environments such as 'HalfLife', 'Quake 3', 'Unreal Tournament 2003' and game engine formats '3D Game Studio', 'Blitz Basic' and 'Dark Basic'.

What all these packages demonstrate is the willingness of the consumer to turn 'developer'. This willingness has also been observed away from the gaming environment [11] with innovation toolkits associated with simulator software being perceived not as added bonuses, but rather as positive, main features of the product itself. This degree of willingness amongst digital immigrants to engage with sophisticated software as developers is an encouraging sign – the immigrants are gaining confidence and going native.

2.1.1 Church Builder in Context

Standard 3D modeling tools such as 3D Studio Max and Maya are generic tools that enable the user to build models of anything they like from geometric primitives components such as boxes, cones, and extruded shapes. Professional users are presented with infinite creative flexibility. School children and teachers are faced with a very steep and arduous learning curve, making the use of the software in schools largely impractical. In fact, as high-end tools face market pressures from mid-range tools in terms of functionality and cost, the functionality of high-end tools becomes increasingly complex, and difficult to master [19]. Although they are, undoubtedly, a lot less complicated to use than 3D Studio Max or Maya, and have made significant design accommodations for non-professional users, packages such as UnrealEd, Truespace and Gamespace are still aimed at a relatively sophisticated user, are specifically purposed for a gaming environment output and, with the exception of UnrealEd, are relatively expensive. Significant advances have been made in repurposing UnrealEd to create Web 3D content [1] but the obstacle of assumed levels of user sophistication remains.

The core design concept of Church Builder has been the creation of a 3D modeling tool which children could "get the hang of" after a brief demonstration and 15 minutes of coordinated practice. The aim has been that the tool should become 'transparent' technology, subsumed in the act of building, just as the technology of pen and paper is subsumed in the urge to set down a story on paper. Consequently, Church Builder appears unique in its simplicity and approach to task accomplishment compared with its peers. Just as the child begins to think, "Now what should I write?' Church Builder aims to stimulate the question, "Now what should I build?"

2.2 INHERIT Project

A similar question simulated discussion within the development team. What kind of new software should *we* build? A tool that could build lots of things? Or a tool dedicated to building one kind of thing? This question was further complicated by the fact that the proposed software development was to take place within the context of a larger project, the INHERIT project. INHERIT aims to allow school children to share in the creation and distribution of cultural heritage models on-line, within the context of a collaborative environment enabled by the Internet. [5]. Self-evidently, in this context, a multi-purpose tool would be advantageous in terms of flexibility of model development. It was decided, however, to develop a tightly focused tool system, dedicated to the modeling of a single kind of thing, a building. The reasons for this decision were threefold:

i. It would hold true to the core of the design concept, which was simplicity. Removing a decision layer from the building process would make the software easier for a child to use.
ii. It is pedagogically practical to have a product without built-in opportunities for distraction. From a practical point of view, the teacher knows that the class cannot be distracted, for example, into building castles when today's class topic is churches.

iii. It would be beneficial to the early development cycle, premised on exploring innovative information visualization techniques, to focus on a single kind of thing. Although it was acknowledged that future developments would centre on plug-ins around a generic core.

Furthermore, it was decided the focus of the tool would be even more closely defined. As the software would be targeted at UK school children, whose subject teaching is determined by the National Curriculum, it was determined to develop a tool specifically designed to assist in meeting the requirements of a specific part of that curriculum. Given the INHERIT context of the development, the target area of the curriculum was set as History at Key Stage 2 [15] with the emphasis on local study, churches in particular. Hence the software is called Church Builder. Churches were identified as suitable objects for modeling as, in many communities, the church will be the oldest, most historical local building. For teaching history, churches serve as a useful focal point as much historical social change is caught up in the history of the building. From a pragmatic point of view churches are geographically close to their communities and so are easily accessible for school visits. Additionally, many faith schools have existing strong links between school and church. From a development perspective, a church comprises of easily identifiable components, systematically assembled, which appear ripe for digital development.

3. CHURCH BUILDER

The resources needed by Church Builder are extremely modest. Church Builder currently runs successfully in primary schools using old PCs with Windows 98 installed on 800 x 600 displays with 32-bit colour set. The speed of the graphics is quite acceptable. Of course it looks better on more modern PCs but, in the UK, not every school can afford them. Installation is very quick - around 10 seconds to be up and running from double clicking on the installation file. Some schools have issues with installing software so, alternatively, the software can be run from a CD in the CD drive. Church Builder has a very small footprint in terms of installation size (of the order of 6 Mb including 2.5Mb of xerces DLL to handle the XML).

3.1 The Building Process

The key feature that allows rapid learning of Church Builder is that the user builds from components relevant to the domain. Thus a church is built, not from boxes, cones and other geometrical primitives but from naves, chancels, towers, porches and sideaisles. The building process enables the user to build the core church, in three clicks, by adding optional nave, crossing tower and chancel components. These components can then be modified by selecting them and changing their core parameters or attributes. The three main components can be further modified by adding items to their walls such as pictures, arches, windows, buttresses, transepts, sideaisles, towers and porches. Walls can be divided into several horizontal sections each with its own set of arches, columns, windows - clerestories are built in this way. Towers are regarded as stacks of tower sections. Tower sections may be square, round, octagonal, spire, or dome.

3.1.1 Visualization of Data

The user has two visualizations available. The 3D view is the main view, which the user can rotate and zoom as building progresses. Components are selected for modification by clicking on them in the 3D view. The 3D view can be interacted with in different modes:

- Edit mode
- Tour jump mode
- Tour glide mode

The first mode is for use while developing the model. The other two enable users to visit their model either by hopping directly between a set of automatically generated way points or gliding smoothly between them. In these tour modes, the user can also move around freely using arrow keys to move forward, back, turn left and right and other keys to look up and down. The user can toggle the 3D view to full screen size to enhance the experience. The 2D view was added to simplify the selection of arches and windows. When a wall is clicked on in the 3D view, it is shown in 2D elevation (and end on 2D elevation) in the two 2D views. Handles are added to the arches, windows (and buttresses) to enable the user to select them. The handles have been recently modified to enable the user to drag arches, windows and buttresses to new positions. Measuring the success of this feature will be the subject of further trials in schools.

3.1.2 Controls

Generally, once an item is selected, its corresponding controls appear. There are two kinds of control that appear when a component is selected:

i. *Context sensitive buttons*: these enable the user to add another component of the same type, delete the component or add an appropriate component of another kind (e.g. add an arch to a selected wall section). Recent additions in response to school trials include buttons to centre items such as arches on a wall and reverse the direction of the arches and columns. This latter is used when completing one side of a church and copying and pasting it to the other side where it needs to be reversed - thus saving a lot of repetitive labour.

ii. *Parameter value controls*: each attribute (e.g. nave length, breadth, apexheight, sidewall height) of the component is represented by:
    a. a label
    b. an edit box for direct entry of parameter values
    c. a slider control for alternative setting of parameter values

Some components such as arches have type attributes, which can be changed from a drop down list of possibilities. Thus arches can be selected to be round, equilateral, lintel etc. Picture objects have a filename attribute that can be used to select jpeg images to place on walls. We have found that children enjoy exploring the possibilities provided by the context sensitive buttons and universally use the slider to set values.

3.2 Hierarchical Data Structures

The developing model is maintained in a hierarchical data structure which, when saved, is expressed as an XML file using tags such as **<nave>**, **<wall>**, **<buttress>**. Parameters are expressed as XML attributes e.g.:

```
<nave length="100.00" breadth="40.00" apexheight="50.00" sideheight="40.00">
```

Components within components are expressed as nested tags e.g.

```
<church>
      <nave>
            <roof>
            <wallleft>
            <wallright>
            <wallback>
      <chancel>
</church>
```

Addition and modification of components is effectively adding nodes to the XML and modifying the node attributes respectively. 2D and 3D rendering is achieved by creating a parallel visual data structure that is to be read by the OpenGL graphics routines that draw the 2D and 3D images. This visual data structure is modified whenever a model component is added, removed or modified. Calculations transform, for example, a nave description that consists of just four real number valued attributes (length, breadth, apexheight and sideheight) into arrays of vertices and normals. Normals are used by the 3D renderer in lighting calculations. Repeated rendering from these arrays involves no overhead of calculation. All of the calculation is carried out when the model itself is changed. Thus rotation, zooming during editing and touring are done without recalculation.

3.3 Church Building from Parameterized Components

Every component added to a church model in Church Builder can be modified by changing its parameters. Components are normally added to walls of existing components. The nave, crossing tower and chancel are exceptions which are treated as starting points. Further components are added by adding to the walls of these three starting points. We illustrate parameterisation by describing the tower component. A tower is regarded as a stack of tower sections. The tower as a whole only has three parameters: width, offset from left hand edge of "owning" wall and insertion (how far it is "inserted" into the owning wall). Each tower section has a height, wall thickness, type (e.g. square, round, octagonal, dome, spire) and an "insert" which determines how much narrower (or wider) it is than the tower section below. Other parameters are determined according to tower section type. For example, octagonal towers have a wall length ratio parameter that gives the ratio of the length of the front side to the length of corner side. Further parameters are available depending upon the types of adjacent tower section. Thus a round tower section sitting on top of a

square tower section requires a support such as a squinch to be inserted so squinch height is an extra parameter for the round tower section. For more on squinches see Section 3.4.2 below. We illustrate towers in Fig 1 and Fig 2. Fig 1 shows a crossing tower consisting of a square base surmounted by an octagonal section and topped off with a dome. Fig 2 shows how a tower with pinnacles can be modelled - the main tower walls have thinner towers added whose top sections are spires.

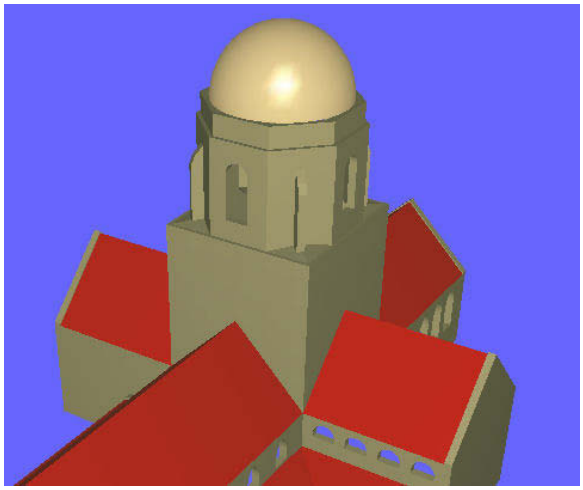

Fig. 1: Tower with octagon and dome.

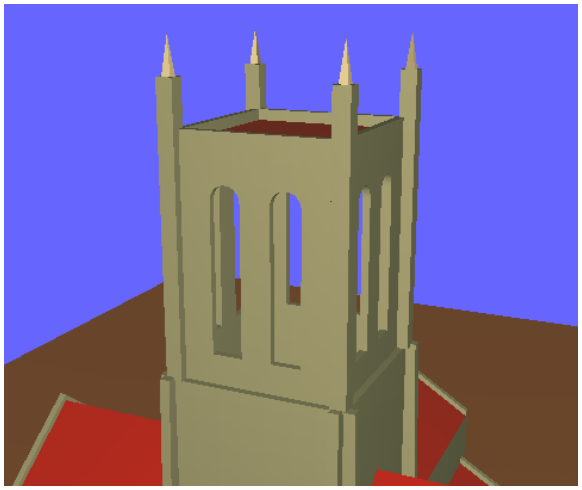

Fig. 2: Tower with pinnacles.

3.4 Church Builder and Computational Geometry

Geometry is incorporated into the model by using the component name to identify how the parameters/attributes are to be used. The derivation of arrays of vertices and normals is based on the laws of geometry. We illustrate this with two examples.

3.4.1 Geometrical Calculation of an Arch.

The arch is regarded as composed of a left hand and a right hand arc of a circle. While this prevents accurate reproduction of Ogees (pointed and usually of four arcs) and Tudor arches (which might be four-centered, parabolic or hyperbolic), arches such as Romanesque/Norman and Gothic [3], [8], can be reproduced accurately. Ogee and Tudor arches will be the subject of future development work.

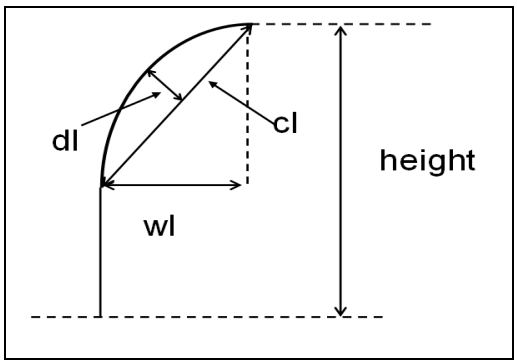

Fig. 3: Measurements for the left side of an arch.

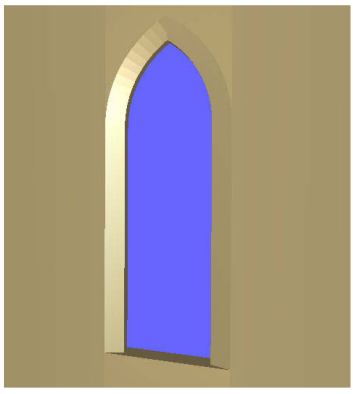

Fig. 4: Equilateral arch with soffit.

Fig.3 shows the left hand side of an arch. The curved part of the arch is specified by parameters cl, dl, wl. There are corresponding parameters cr, dr and wr for the right hand side. The values of dl, cl and wl can be measured either

directly on site or else using photographs. Church Builder uses their values to calculate the radius and centre of curvature of the arc and hence the vertices and normals that enable its visualization as part of an OpenGL quad strip. The general "arch" arch type enables the user to modify left and right independently enabling asymmetrical arches to be produced. In contrast the "round arch" type will just use the width of the arch to determine the six parameters for the left and right arcs. All the user has to set is the width. Similarly, the "equilateral arch" type consists of two arcs each centred at the springer of the other arc. Thus, again, the user only needs provide the width and the six parameters can be calculated. In the case of the "lintel arch", which is in fact just a rectangular opening, cl, wl, cr, wr are each equal to half the arch width while dl and dr are zero. The user only specifies the width. The arch soffit is parameterised by two values that determine its shape. The effect is demonstrated in Fig. 4. Adding moulding to the soffit will be the subject of further work.

3.4.2 Geometrical Calculation of a Squinch.

A squinch is an arch or system of concentrically wider and gradually projecting arches, placed diagonally at the internal angles of towers to fit a polygonal or round superstructure onto a square plan [8]. Without the support provided by squinches, the superstructure would be unsupported at the corners and so liable to collapse. Church Builder automatically includes approximations to squinches when the user builds a tower in which a round tower section or dome is placed on top of a square tower section and when an octagon is placed on top of a square section. A more sophisticated development is the pendentive; which is a triangular section of a sphere. A good example of a pendentive can be seen supporting the dome of the Hagia Sophia in Istanbul. Church Builder's implementation of the squinch will be refined to include arches and extended to produce pendentives in future versions.

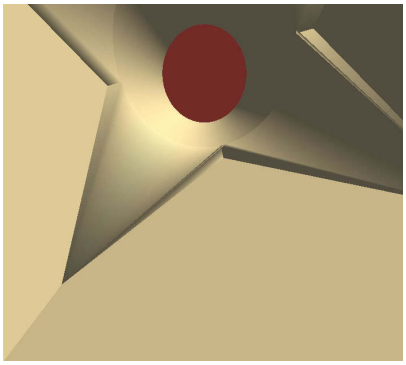

Fig. 5: Squinches under a round tower section.

Fig. 5 shows squinches that have been automatically inserted under a round tower section sitting on top of a square tower section. The squinch follows the curved edge of the round tower section at the top and tapers down to a single point some distance below. The user only sets the distance of the bottom point from the level of the base of the round section - everything else about the squinch is generated automatically from the geometry of the two tower sections.

3.5 Parametric Constraints

An important aspect of the geometry is the application of constraint**s**. Every parameter has minimum and maximum parameter values set on the parameter sliders when the component is selected. Some of these are reasonable but arbitrary, for example the nave length maximum is determined by using the largest known nave size ever used. Others depend on the geometry. For example, a porch on a wall cannot have an offset from the left end of the wall that takes it outside the wall. Similarly, an arch cannot be moved outside the wall it is contained by. In the case of a gable end, an arch cannot be moved so that its top lies outside the gable. Thus, in Fig. 6, we can see the arch at its rightmost extent. Notice how the sliders for left offset, height and width are at their maximum. In this position the arch cannot be moved further left, or made taller or wider without going outside the wall and Church Builder knows this.

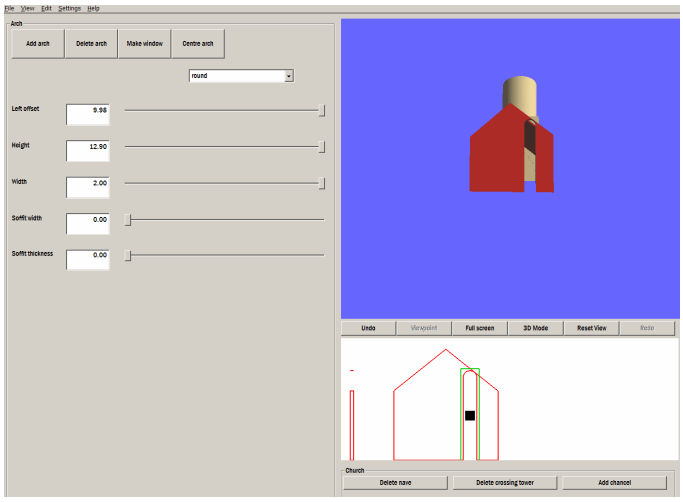

Fig. 6: Arch at its rightmost extent.

Similar constraints apply to attempting to make a wall containing arches, windows or with porches, transepts etc shorter in length. Church Builder does not allow the wall to be too short to contain the arches and windows or become detached from the attached porches and transepts etc**.**

4. CHURCHBUILDER FUNCTIONAL CHANGES

4.1 Not so Wonderful Wizards

The initial view of the developers was that a Wizard approach to building would be the most appropriate approach[6]. Churches are built systematically so why not have Church Builder create models systematically? The first version of Church Builder identified systematic building as consisting of the following sequence with each step becoming a page in the Church Builder Wizard:

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

To the developers this appeared to be a natural way to build a model consisting of components. Unfortunately, this view was not one shared by the children in the first Church Builder trial. For them, having built the nave, the next step was, invariably, to add doors and windows. They were disappointed to discover they had to wait until page 6 to do this! On reflection, the developers recognized that the sequence imposed by the wizard was an artificial sequence. Real churches are built from the foundations up, windows and all. In the medieval period larger churches and cathedrals would be built section by section, perhaps the chancel first, then the nave then the transepts. In response to the trial observations the wizard was removed and replaced by a single dialog box, allowing users to build in own their preferred order.

4.2 Tree Surgery

The single dialog box, which replaced the wizard, initially included a tree control that children could use to select components to modify, instead of selecting components in the 3D view. In trials, children seemed au fait with this process. Teachers appeared bemused. It became clear that to accommodate the needs of the gate- keepers the tree had to go, and that component selection in the 3D view was the way forward. This decision was also ergonomically advantageous as it resulted in a less cluttered interface.

4.4 Safety Nets and Go Faster Facilities

Early on in the development cycle the provision of Undo / Redo facilities was recognized as an essential design element. This facility acts as a 'psychological safety net' for users, encouraging experimentation, safe in the knowledge that unintentional changes can easily be 'undone'. In addition to supporting user experimentation this facility also enables teachers and facilitators to 'reverse' the children out of tight spots and set them back on course. This is reassuring for teachers as it requires no in depth knowledge of the software to swiftly set problems right.

A significant improvement to the development model has been the provision of Copy / Paste facilities. This allows, for example, complex towers to be constructed and immediately copied and used elsewhere in the model. This facility is satisfying for users as it speeds up workflow and facilitates the production of otherwise labour-intensive, highly complicated, models. This facility is also reassuring for teachers as it mimics facilities and processes available in software with which they are already familiar and assists memory for routines by promoting recognition rather than recall in the interaction.

4.5 Church Builder Interface Changes

Paradoxically, the quest for a simple, single level information architecture during the initial development of Church Builder had resulted in a cluttered and idiosyncratic interface, which was confusing to use [Fig. 7], [Fig. 8], [Fig.9]. A re-design was needed to increase usability and maximize the ergonomic efficiency of the interface. Primarily this consisted of 'zoning' the screen into 'work' areas and 'viewing' areas and creating a multi-layered information architecture, which mapped onto existing mental models of the target users [Fig.10] [Fig 11]. This reorganization simplified the existing interface and laid information architecture foundations, which anticipate future development.

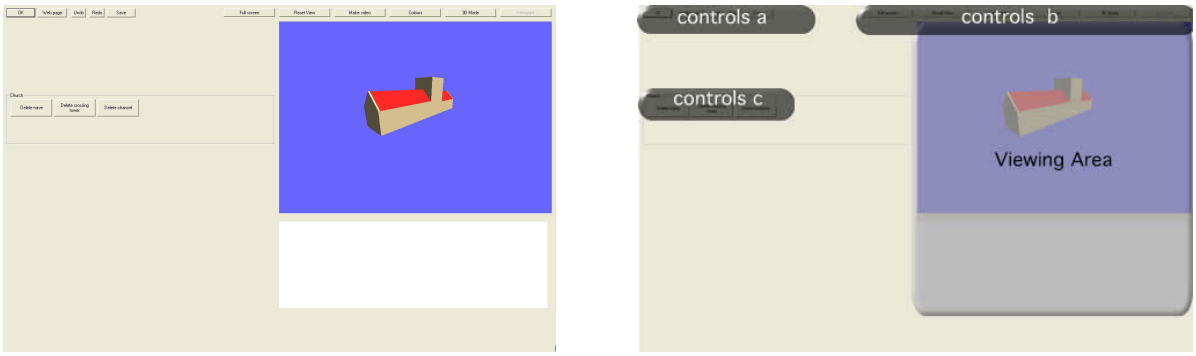

Fig. 7: Interface before redesign with confusing functions layout. Control groups as set out in Fig. 8.

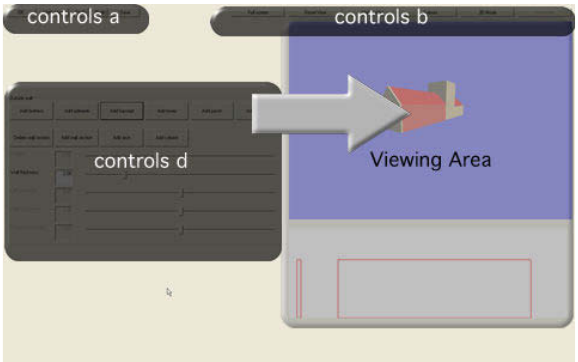

Fig 8: Interface,before redesign, once initial nave, chancel and crossing tower have constructed and an element has been selected in the viewing window. Notice Group C controls from Fig. 7 have vanished. Details of control groups set out in Fig. 9.

Group A Controls	
OK	
Web Page	Make a web page displaying the current model
Undo	Undo last alteration to model
Redo	Redo last alteration to model
Save	Save model
Group B Controls	
Full Screen	
Reset View	
Make Video	
Colours	
3D Mode	
View Point	
Group C Controls	
Add/Delete Nave	Add or remove nave
Add/Delete Crossing Tower	Add or remove crossing tower
Add/Delete Chancel	Add or remove chancel
Group D Controls	
Parametric controls	Element specific parametised controls, in this instance relating to the selection in the viewing window of an outside wall

Fig. 9: Control Groups of interface buttons as set out before the interface change. Notice the erratic grouping of edit, viewing and file save controls.

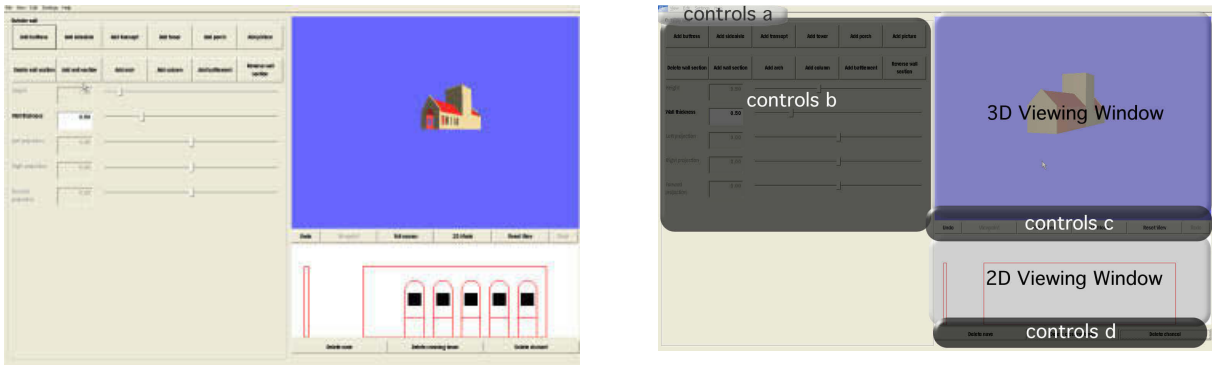

Fig. 10: Interface after redesign, Control groups as set out in Fig. 11.

Group A Controls	
File	
View	
Edit	
Settings	Set colors of display
Help	Help Menu
Group B Controls	
Parametric controls	Element specific parametised controls, in this instance relating

	to the selection in the viewing window of an outside wall
Group C Controls	
Undo	Undo last alteration to model
Viewpoint	
Full Screen	
3D Mode	
Reset View	
Redo	Redo last alteration to model
Group D Controls	
Add/Delete Nave	
Add/Delete Crossing Tower	
Add/Delete Chancel	

Fig. 11: Control Groups of interface buttons as set out after the interface change. Notice the regrouping of functions and more ergonomically efficient placement of groups. Group D controls remain permanently available in the redesign and Group A controls take advantage of existing mental schema of function groupings in windows software.

5. CONCLUSIONS

Church Builder has proved itself capable of delivering a fast, effective and fun CAD tool to a user- population, which, hitherto, had been considered incapable of building sophisticated 3D models. It is a significantly better 'tool for the job' than existing complicated and expensive alternatives and has been met with enthusiasm, even in the early stages of development. The parametric approach to modeling is efficient and the information architecture of the revised interface has significantly improved on the development model. The newly revised interface overcomes the 'teacher as gatekeeper' obstacle by mapping the information structure of Church Builder on to similar existing mental models. The interface needs further work to address specifically 'child target user' usability issues. Key to improving child usability is the migration to an aesthetically pleasing GUI interface and considered use of colour [18] both in the interface and in the model itself, which is the next anticipated stage of development. Currently, Church Builder aims to enable children to model real churches and so provides a palette of model colours confined to greys and browns to provide rough approximations of the materials from which churches are constructed. Future development will add texture facilities in which images of stone and brick can be "painted" onto the walls to make more realistic effects. Similarly lead, slate, tiled or thatched roofs will be made available. It is anticipated that, in addition to 'realistic' options, children would also want the facility to use brighter colours of their own choosing. In addition to the user satisfaction gained from making such a facility available, the developers recognize that there is also an historical justification for making such a facility available. The older, grey and brown pre-Reformation English Churches we see now are not presented as they would once have been. When originally built as Roman Catholic churches, the liturgy they housed and the theology they expressed encouraged painted decoration in a manner that modern tastes might find garish [7],[12], [20]. Ironically, children painting their churches from a 'secular' palette could well be painting them as their original builders intended.

6. FUTURE WORK

The developers are currently collaborating with educationalists to create teaching materials in which Church Builder will be centrally embedded. Thus the teacher will have available the software and full lesson plans and support materials to deliver the National Curriculum in an way that excites and motivates the children.

7. REFERENCES

[1] Arendash, D.: The Unreal Editor as a Web 3D Authoring Environment, Proceedings of the Ninth international Conference on Web 3D Technology, ACM Press, New York, NY, USA 119-126, 2004.

[2] Campbell, C.; Campbell, B.B.; Dickinson, D.: Teaching and Learning through Multiple Intelligences, 3ed, Allyn and Bacon, Boston, USA, 2004.

[3] Corkhill, T.: A Concise Building Encyclopedia, Pitman and Sons Ltd., London, 1946.

[4] De Aguilera, M.; Mendiz, A.: Video Games and Education (Education in the Face of a "Parallel School"), ACM Computers in Entertainment, 1(1), Article 1, October 2003.

[5] Farrimond, B.; Hetherington, R.: Compiling 3D Models of European Heritage from User Domain XML, IV05, Proceedings of the Ninth International Conference on Information Visualization, IEEE Computer Society, Los Alamitos, California, 2005.
[6] Farrimond, B.; Hetherington, R.: Developing a Parametric approach for 3D Modeling Software, IV06, Proceedings of the Tenth International Conference on Information Visualization, IEEE Computer Society, London, England, 2006.
[7] Fletcher, B.: A history of Architecture (18th ed.), The Athlone Press, University of London, 1975, 674.
[8] Fleming, J.; Honour, H.; Pevsner, N.: A Dictionary of Architecture (2nd ed.), Penguin Books Ltd., England, 1972.
[9] Gamespace, http://www.caligari.com/gamespace/intro.asp, Caligari Software.
[10] Gardner, H.: Intelligence Reframed, Multiple Intelligences for the 21st Century, Basic Books, New York, NY, 2000.
[11] Henkl, J.; Thies, S.: Customization and Innovation – User Innovation Toolkits for Simulator Software, Proceedings of the 2003 Congress on Mass Customization and Personalization (MCPC 2003) Munich, 2003.
[12] Long, E. T.: Screen paintings in Devon and East Anglia, Burlington Magazine for Connoisseurs, 59(343), pp 168-171and 174-176, October 1931.
[13] Microsoft: Multiple Intelligences and ICT Learning, Microsoft Education United Kingdom. http://www.microsoft.com/uk/education/learning/multiple-intelligences.
[14] Prensky, M.: Digital Natives, Digital Immigrants, On The Horizon, 1(5), 2001.
[15] The National Curriculum Online, History, Key Stage 2, http://www.nc.uk.net/nc/contents/Hi-2--POS.html.
[16] Truespace, http://www.caligari.com/gamespace/intro.asp, Caligari Software.
[17] UnrealEd, http://udn.epicgames.com/Two/IntroToUnrealEd, Epic Games.
[18] Lavie, T.; Tractinsky, N.: Assessing Dimensions of Perceived Visual Aesthetics of Web Sites, International Journal of Human-Computer Studies, 60(3), 269-298, 2004.
[19] Unver, E.: Strategies for the Transition to CAD Based 3D Design Education, Computer-Aided Design and Applications, 3(1-4), 2006, 323-330.
[20] Wells Cathedral, http://www.bbc.co.uk/history/british/middle_ages/launch_gms_paint_wells.shtml