THE USE OF 3D VISUALISATION AND INTERACTION TO MARKET A NEW NEAR-NET-SHAPE PIN TOOLING APPLICATION

Kevin Badni
Loughborough University, Department of Design & Technology, Loughborough, U.K.

Keywords: 3D, collision detection, STL parsing, interaction, Director, Pin Tooling.

Abstract: This paper describes the methodologies used to create an interactive 3D multimedia application. The application was required to fulfil a business need of representing as close to reality as possible, how a new product technology worked. A number of IT solutions for the application were researched and described within this paper. The construction of the application is then discussed with the methodology behind a stereolithography parsing system and the algorithms used to detect collisions being described in detail. The application has been successfully implemented by the business to build an engaged client basis.

1 INTRODUCTION

Surface Generation is a privately owned UK company which has a revolutionary Pin Tool, mould making, process with intention to 'fundamentally alter the economics of small volume production of large products'. (Surface Generation 2007).

Figure 1: Surface Generation's Pin Tooling. Copyright Surface Generation 2004-2007.

The Near-net-shape Pin Tooling (NPT) technology that the company wanted to promote uses beds of adjustable square pins to create near net shape surfaces (see Figure 1). These beds are designed for use as a support structure for composite tooling systems. Their approach has been designed to reduce lead time and labour issues which are at the centre of large component manufacture. Due to the unique nature of this product Surface Generation wanted to use information technology as a marketing device to demonstrate how their unique technology worked.

The proposed application was to allow the user to 'play' with the technology and so understand by experience how the technology worked and how the users decisions could influence the final results. The hypothesis that interacting with a virtual product would allow for a customer to gain a better understanding of the technology involved is backed up by existing research as described by Schlosser (2003) 'The act of directly manipulating a virtual object should produce clearer mental images than if this information were acquired passively, regardless of the user's goal'. This is supported by Rosenblatt (1965) who describes mental imagery as being a critical aspect of information processing. The proposed application would be designed to mirror the real life technology with the intention to persuade the customer that the new technology is a feasible option for them to purchase. To facilitate this Biocca (1992), states that in theory the application should "blur the "boundary between simulation and communication, and virtual and physical reality". As discussed by Kahneman and Tversky (1982), the easier and more vividly individuals can envision a scenario, the higher the likelihood that they will estimate that the scenario will actually occur.

Badni K. (2008).
THE USE OF 3D VISUALISATION AND INTERACTION TO MARKET A NEW NEAR-NET-SHAPE PIN TOOLING APPLICATION.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - HCI*, pages 325-330
DOI: 10.5220/0001678403250330
Copyright © SciTePress

This technique of representing the experience by using vivid mental imagery rather than expecting the customer to use cognitive elaboration has been proposed to be a superior method in influencing intentions (MacInnis and Price 1987). Whilst the creation of an application that requires little cognitive elaboration is ideal, it must be acknowledged that a web based 3D design is always a compromise between realism and what can be displayed considering computer software, hardware and bandwidth. As stated by Davis (1996) complete imitation is impossible as it is inevitable that some process of subtraction and modification of representation is always required. Nevertheless from a marketing perspective it can be argued that the more a customer can envisage themselves using a technology, the more influential it will be in affecting their expectations to purchase.

2 INFORMATION TECHNOLOGY OPTIONS

The author was asked to research into possible IT solutions which could sit on Surface Generation's website. The application was to be up and running within a tight deadline of only three months. The IT application required specific actions to be undertaken which generally mirrored the real-life process used by Surface Generation, these are set out in Table 1.

Table 1: Application Process.

Actual System	Virtual System
Load STL files	select STL file
	create 3D model
pin bed reset	create pin bed
run collision detection	run collision detection
position pins	position pins
create infill	create mould volume

To facilitate these operations a number of approaches were considered. As the application was to represent the real-life technology and hardware so reduce the need for cognitive elaboration, a 3D virtual model would need to be created. Due to the application revolving around a virtual 3D model the first obvious IT choice to be considered was VRML (Virtual Reality Mark-up Language).

2.1 VRML

VRML is a world standard for web based virtual reality interfaces. VRML is a scene description language based on a subset of Open Inventor from Silicon Graphics that describes the geometry and the internal behaviour of a 3D scene. Even though VRML is not a general programming language it does have some features that can make it a useful tool for dynamic 3D environments. These features include the ability to write prototype nodes which can incorporate scripts to perform various behaviours within the VRML world. With the creation of this standard a number of dedicated web browsers have been produced which are used to display and control virtual worlds using VRML over the internet. These browsers are cross platform compatible and interpret the VRML files displaying a 3D model within the browser (Badni 2005).

Having investigated the use of VRML as an application base, a number of negative issues were raised. VRML is mainly used to visualise 3D models, it is not normally used to facilitate heavy computational tasks such as the required collision detection. VRML requires a dedicated browser to be used, so the customer would have to download and install a browser. VRML browser's rendering engines would also struggle with the number of polygons used within the imported STL (stereolithography) files, making the updated frame rate rather slow. The frames per second rate (FPS) is affected by the processor speed, graphics card and memory available on the customers PC. As this is unknown, the 3D designer must design the 3D scene to be as computationally light as possible yet still maintain a high level of model realism. The 3D model files would therefore have to contain the minimum number of polygons (facet shapes that make up a virtual model) as possible. However as the application would use complicated STL files with a large number of polygons (>2500) this would not always be possible. Having undertaken some experiments it was clear that having more than 2,500 polygons displayed would reduce the frame per second rate below the acceptable 10fps rate. Hartman and Wernecke (1996) recommend that the designer should aim at achieving a minimum frame rate of 10 FPS, which is the frame rate required to simulate visual continuity whilst moving in the virtual world.

2.2 Java

The second IT option was to create a bespoke application using a high level programming language such as Java which was formalised in 1995. Because the Java language is an interpreted language, it requires an interpreter (also known as a Just-in-time compiler) in order to run, so as long as the customer has 'The Java Virtual Machine' on their PC, the application will theoretically run. The main advantage that a proposed Java application had, was due to the nature of Java programming the computational side of the application could be relatively easily handled. The major downside of creating a bespoke application using Java is the enormous time scale that would be involved. Surface Generation wanted a working application within three months which ruled out a bespoke application being created by just the author.

2.3 Director

The third option, and chosen route was to use Macromedia's Director authoring environment. (now owned by Adobe). Director has an object-oriented development environment, similar but not as powerful as Java but much more powerful than VRML. Director also has a pre built 3D rendering engine so can handle 3D model files. The other advantage Director has is the common usage of the Shockwave plug-in, as over 200 million web users (58.5% of mature markets, Millward Brown 2007) having already installed the Shockwave Player.

3 APPLICATION DEVELOPMENT

3.1 Reading STL Files

The first stage of the application was the reading of imported STL files. The real life NPT application uses customer's STL files to set up the pin positions. This was to be mirrored in the virtual web application by selecting from a library of different STL files. An STL file is a triangular facet based representation that approximates surface and solid forms. An STL file as described by Burns (1993) 'consists of a list of facet data. Each facet is uniquely identified by a unit normal (a line perpendicular to the triangle and with a length of 1.0) and by three vertices (corners). The normal and each vertex are specified by three coordinates each, so there is a total of twelve numbers stored for each facet.'

The standard STL format specifies that the object represented must be located in an all-positive octant. In other words, all the vertex coordinates of the facets must have positive numbers. However, with a few exceptions most STL creation software allows the facets to be in any arbitrary location in 3D space. Director also works within a 3D space, however Director's co-ordinate system is unfortunately incompatible with an STL file so ruling out a straightforward direct import. Therefore, a new STL parsing system had to be written to interpret the STL models into the Director environment.

The first part of the parsing system involved stripping out each individual vertex (3D points) position from the ASCII STL text file. The idea then was to recreate the mesh using Directors own rules. Within Director a vertices value needs to be entered in to the code that will indicates the number of entries that will be expected in a linear list of vectors, each of which describes the location of one vertex within the 3D model. This vertex list then has to be cross referenced with individual facets in order to work out the connections between the vertices that would create the final triangles. To achieve this with the different STL models available from the library a new algorithm had to be created that could sit within a repeat loop within the application to determine which faces from the STL file were based on which vertices. The method undertaken, involved creating a breakdown list of a number of sample STL surfaces, (see Figure 2 and Table 2), listing each vertex which related to each face. A pattern emerged for both the odd numbered faces and the even numbered faces, these have been described below.

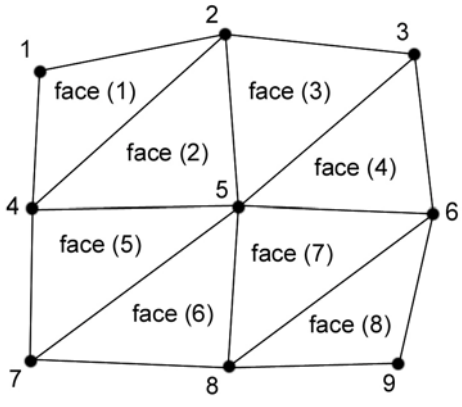

Figure 2: Facet STL surface.

Table 2: Vertex List.

Face Number	Vertex 1	Vertex 2	Vertex 3
1	1	4	2
2	2	4	5
3	2	5	3
4	3	5	6
5	4	7	5
6	5	7	8
7	5	8	6
8	6	8	9

```
Even Faces = [Vertex1,Vertex1+the
columns+1,Vertex1+1]
Odd faces = [Vertex1+1,Vertex1+the
columns+1,Vertex1+the columns+2]
```

However a discrepancy was found with vertices that were on the edge of a surface (i.e. Vertex 6) . To circumnavigate this issue the arithmetic modulus operator was used to check if the vertices was on an edge:

```
If Vertex1 mod (columns+1) = 0 then
Vertex1=Vertex1+1
End if
```

This would only come into effect if the STL file created was not fully closed, otherwise in theory there would be no edge vertices.

Once the vertex position and connections had been resolved they were arranged and positioned in the 3D space in a counter clockwise order when looking at the triangular face from the outside using the right-hand rule with the thumb pointing up being 'front' (see Figure 3). This rule was applied to make sure the 'front' or back' side of the triangle was correctly orientated.

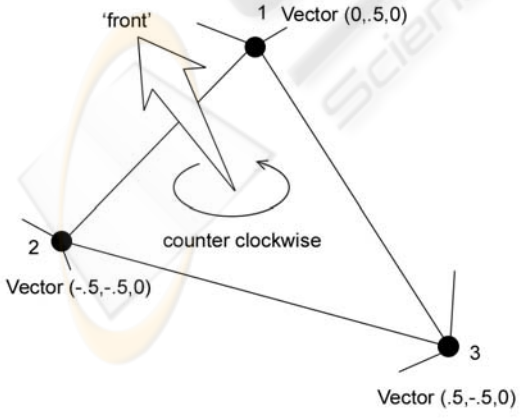

Figure 3: Front face of facet.

Within Director the Camera property has a front direction vector, any normal that is between 0 and 90 degrees of that vector will be drawn with the front face, if it is beyond 90 degrees the back face is drawn. If the back face were drawn the 3D STL model would appear to have holes in it and would not work with the collision detection scripts. Once the faces had been drawn correctly the final step was to perform a relative scale on the model so that it would fit within the pin bed footprint. This was due to the fact that STL files do not contain any information on scale as STL units are arbitrary.

3.2 Creation of Pin Beds

The second step was to allow the customer to create a number of different pin beds. There were two options available within Director to create the beds. The first was to build the beds in an external 3D modelling environment and then import them into Director. This had the advantage that the different beds could be quickly called upon with the 3D environment, however it had the downside increasing the applications size so increasing download time. With this is mind the other option was chosen, that of making the pin bed from within Director itself (see Figure 4). This had the advantage of adding very little size to the application only lines of code.

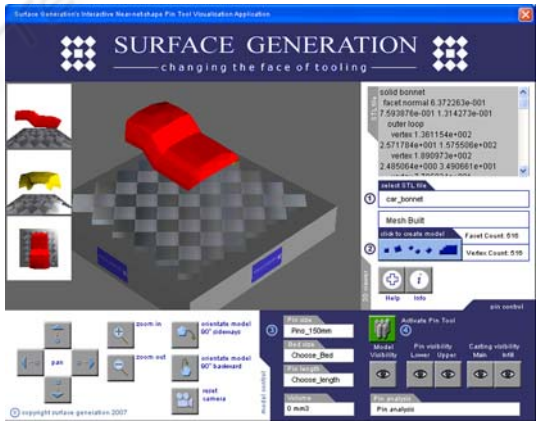

Figure 4: Creation of Pin Beds.

3.3 Collision Detection

Once the 3D model was in place and the pin bed had been built, the next stage was to move the pins in a vertical vector to their Near-Net-Shape positions. To facilitate this, it was necessary to work out which pins would hit the model and at what vertical position this would happen. It was originally

conceived that due to having built a vertex list for the STL models, a simple comparison of vertex positions to pin positions could have been used. However, the STL model could be re-orientated and repositioned within the Director application so rendering the list irrelevant. The use of collision detection was implemented instead. Due to the high computational requirements of collision detection, it inevitably incurs a 'cost'. Costs relating to collision detection are described by Phelps and Cloutier (2003) as being a 'certain level of computational complexity relative to the level of accuracy in detecting collision. Generally speaking, the more precise the accuracy of a given methodology, the more computationally intensive the calculation, and the larger the drain on the overall program'. To help reduce this cost a number of algorithms were used simultaneously within the Director environment.

The first algorithm created a bounding sphere around the 3D model, to be used to see which pins were directly below the sphere. This was created by taking the pixel furthest from the centre of the model and using the distance vector as a radius. This was a quick way to see if the sphere was directly over a pin, if the returned result was negative, then the pin was ignored, if it was true, the pin was added to a 'hit list' of pins that required finer detection work.

The second algorithm undertook the finer detection work. Taking each of the hit list pins a form of spatial partitioning was applied to their top faces. The partitioning split the pin head once horizontally, and once vertically, dividing the space into four quadrants of equal size. If the 3D model was detected within one of the quad regions, that region was then further subdivided into four more quadrants providing a double-level detection zone (see Figure 5). For the larger 150mm pins a third quad level was introduced to improve accuracy. The distance from the pin to the 3D model was then held in an array and overwritten if another part of the 3D model was closer to that pin, ultimately returning the lowest collision distance. This algorithm was then repeated for all lower and upper hit list pins to allow for the next step of the application to take place.

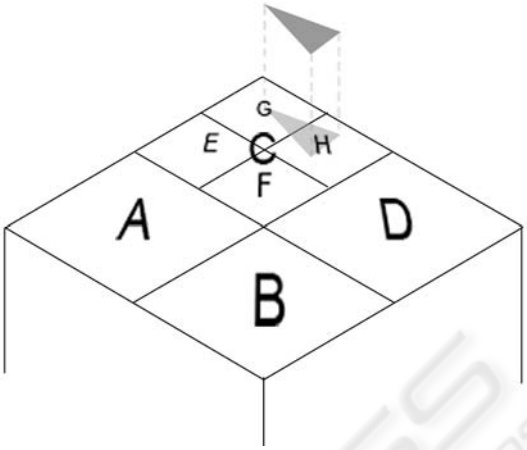

Figure 5: Quad Partitioning of Pins.

3.4 Pin Movement

Once a lower bed and upper bed collision distance list had been collated each pin was moved accordingly to be in contact with the model. The space left was then in-filled with a moulding compound, illustrated within the application by a solid infill block (see Figure 6). The volume of this infill was then calculated and shown to the user illustrating the volume of moulding material they would need in the real world. This fulfilled the original brief of allowing the customer to affect the volume calculation outcomes.

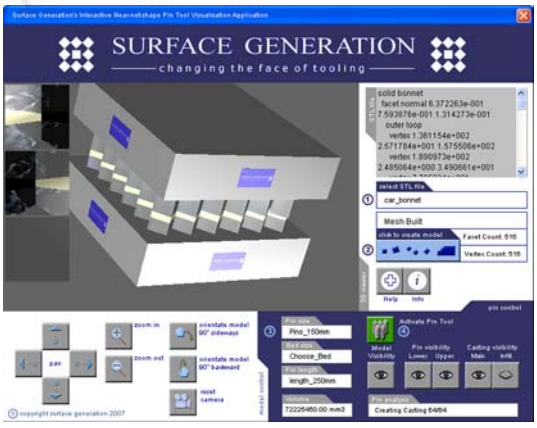

Figure 6: Solid Infill (Lighter Colour) Between Upper and Lower Pin Beds.

4 CONCLUSIONS

The creation of an interactive 3D product simulator was undertaken using the most appropriate technology available to the author. A number of options were examined for each stage of the development, which culminated in a single application which can be viewed over the web by any customer who has the Adobe shockwave viewer.

Research has shown that interacting with a virtual product rather than being exposed to only passive information is a superior method in influencing customers intentions. Having a 3D environment is particularly beneficial to those customers who lack knowledge of the product or are unable to vividly imagine the technology as it does not rely on individuals' self-generated mental images.

The application is presently being piloted by Surface Generation who have currently received positive feedback on the application. This has enabled them to progress their commercial leads with these companies, so fulfilling the marketing objective.

It would be interesting to examine the effect of having been exposed to virtual interaction with customers post consumption satisfaction. As the customer would already be familiar with the capabilities and options of the technology, would the real-life technology be easier to use and manipulate? However it may also result in unrealistic expectations, so making Surface Generation's Near-net-shape Pin tooling system more complicated and frustrating to use. This would be an interesting area for further investigation.

REFERENCES

Badni, K.S. (2005), "Virtual reality design techniques for web-based historical reconstructions", Virtual Reality, 9(4), 1st April 2005, p2.

Biocca, Frank (1992), "Communication within Virtual Reality: Creating a Space for Research," Journal of Communication, 42 (Autumn), 5-22. p 8

Burns, M (1993). Automated fabrication: Improving productivity in manufacturing. PTR Prentice Hall.

Davis,S.B. (1996) The DESIGN of Virtual Environments with particular reference to VRML Report for the Advisory Group on Computer Graphics, Middlesex University.p4.p61-65

Hartman, J and Wernecke, J (1996) The VRML2.0 Handbook, building moving worlds on the Web, Silicon Graphics, Inc. p226

Kahneman, Daniel and Amos Tversky (1982), Judgment under Uncertainty: Heuristics and Biases, New York: Cambridge University Press.pp130-154

MacInnis, Deborah J. and Linda L. Price (1987), "The Role of Imagery in Information Processing: Review and Extensions," Journal of Consumer Research, 13 (March), pp473-491.

Millward Brown (2007), [Online]. Available: http://www.adobe.com/products/player_census/methodology/ [2007, November 25]

Phelps, A. and Cloutier, A. "Methodologies for Quick Approximation of 2D Collision Detection Using Polygon Armatures" by A Phelps and A Cloutier. Published both at the Directors Online User's Group [DOUG] and the Macromedia DevNet forum. 2003

Rosenblatt, Louise (1965), Literature as Exploration, New York: Modern Language Association. Carbondale: Southern Illinois University Press. pp174-178

Schlosser, Ann E. (2003) "Experiencing Products in the Virtual World: The Role of Goal and Imagery in Influencing Attitudes versus Purchase Intentions" The Journal of consumer research [0093-5301] vol:30 iss:2 p:184

Surface Generation 2007 [Online]. Available http://www.surfacegeneration.com/iexplorer/history.htm [2007, November 25]