Digital artworks: bridging the technology gap

Digital Artworks: Bridging the Technology Gap

Colin H C Machin
Department of Computer Science
Loughborough University
Loughborough
Leicestershire, LE11 3TU, UK
+44 1509 222683
C.H.C.Machin@lboro.ac.uk

Abstract

In a drive to produce installation artworks, particularly for public viewing, that are more appealing to the viewer, artists are increasingly turning to "the digital world". Whilst the technology behind such artworks is well established, being commonly found in controllers for industrial machines, the software engineer who provides the firmware strives to make the technology more accessible to the artist. What is required, during the design stage, is an interface that will allow the artist to visualise the artwork and its operation. This paper describes the technologies and the way in which they are made accessible to the artist, demonstrating a software-based simulator built for a particular artwork. It then poses questions for the future, through which further demands for collaboration can be met without compromising artistic creativity.

Keywords: Digital artwork; art and technology; creative collaboration; creativity and cognition; visualisation; simulation.

Introduction

Artists who aspire to using digital techniques in their installation artworks will usually need to seek the assistance of Electronics and Software Engineers. The kind of artwork that is under consideration here is that which has some physical, usually three-dimensional form, such as a sculpture or even part of a building. This category excludes artworks that are portrayed upon a monitor screen or on film or video. Putting aside the simplest artworks of this type, such as Martin Creed's 2001 Turner Prize winning exhibit [1], which involved turning a light on and off, any artworks that portray motion or react to their environment [2] require controlling electronics and software. Endowing the artwork with the required level of control is best achieved through the use of some kind of computer system. A standard approach would probably involve a PC, or at least its internal components, perhaps programmed in a language such as *Visual Basic*. However, the topology of certain target systems may render this kind of solution inappropriate or overly expensive. For example, where a distributed control system is considered to be a preferred option, the work could be delegated to a myriad of controllers based upon smaller and cheaper components. Equally, space constraints may mean that a much more compact solution is necessary. In such cases, it is the job of Electronics and Software Engineers to collaborate with the artist in order to identify and provide the two elements, hardware and software, that will bestow the required degree of control [3]. The Electronics Engineer must provide the computer-based hardware that will control the artwork and, if the artwork is to respond to its environment, the sensing electronics also. The techniques utilised to design and build a controller based upon microprocessor technology are well established and are generally merely simple manifestations of the type of solutions applied to industrial control or automotive applications. Once the hardware is available, the Software Engineer can provide the embedded software required to control the artwork.

Controlling the Artwork

Controlling an artwork of this type is fundamentally no different to controlling an industrial machine. In a very simple structure, the software that is embedded into the controller for the piece simply manipulates outputs on the hardware, which are connected to real-world elements, for example Light-Emitting Diodes (LEDs). In a simple piece, in which LEDs are embedded in a footpath or as part of a building, they could be illuminated in some kind of sequence in response to the passage of time and/or external stimuli. In the first of

Proceedings of the 20th Eurographics UK Conference (EGUK'02)
0-7695-1518-5/02 $17.00 © 2002 IEEE

these two cases, the piece follows a fixed sequence whose timing is predefined, much like a set of traffic lights at a busy junction. Where sensors are involved, the piece appears to be responding in an intelligent manner to the stimulus from one or more given sensors. In fact, what is happening is that the embedded software is simply mapping inputs to outputs according to some, probably fixed, function that couples the two. This behaviour is also seen in traffic lights. When a vehicle on a minor road arrives at an intersection it triggers the traffic light controller to enter the sequence that allows it egress on to the main road. In describing such systems, we can borrow terms from the world of industrial control systems to distinguish these two types of behaviour, although the terms are not used in their strict senses in the current context. We can assign the term *open-loop* to those artworks that simply perform fixed sequences and *closed-loop* to those that sense their environment and make some response based upon what is detected. In any event, an open-loop system is easier to construct. When dealing with a closed-loop system supporting an artwork, the problems that arise are not solely confined to its specification. It is in such systems where we find the greatest challenges in even identifying what the artist requires.

Models for Collaboration

There are many models for describing the style and level of collaboration between technologists, as a group, and artists engaging in installation artwork [4]. This collaboration is not limited to Electronics and Software Engineers, though. Where large structures form part of the piece or where the piece is to be incorporated into a building, the artist will inevitably consult mechanical and civil engineers. At one extreme, the technologist must be prepared to simply undertake to provide the artist with exactly what is required and no more. In this case, the technologist could well be working in something of a void, probably being unsure of the real purpose or outcome of the software effort. An interesting parallel can be drawn here with artists who create pieces within the wider field of the performing arts. A composer will have no more than a mental image of how a conductor will interpret the score and how the orchestra will perform the piece. There is potential for the artist to be disadvantaged too, being in much the same position, unaware of precisely how the Software Engineer will interpret the requirements. In a collaboration that could be described more as a partnership, the technologist will assume an advisory role in which the artist will be made aware of the possibilities of the technology within the framework of the desired goal. In this case, the goals could be considered as shared. At the other extreme, the artist will challenge and stretch the technologist. This could result in the latter coming up with novel ways of solving traditional problems within the technologist's own domain. Here, the artist has acted as a valuable catalyst to the possibly substantial development of the technology.

In any event, though, the decision on how the artwork operates must ultimately remain with the artist and in most cases the ultimate evaluation of the outcome will be solely within the artist's domain. It is in realising this that the Software Engineer suddenly understands that the job is not as simple as at first may have been apparent. It has already been acknowledged that the hardware is at the simple end of the spectrum of control systems. It is almost certain, though, that this is where the simplicity ends.

In the general world of programming, experienced Software Engineers are conscious of the fact that at the stage at which requirements are specified, many clients will be unaware of what is really required. In many cases this is simply due to that fact that the client does not really understand what is possible within the technologist's domain. When working with artists, it is probably here where the biggest challenges arise. Whilst they will clearly possess the necessary creative talents, artists must not be expected to be sufficiently familiar with the world of electronics and computing to be able to make complete and informed design decisions. Clearly cost plays an important part in all of this too, but it is when the collaboration is in the form of a partnership that the client can expect to see the greatest returns on investment. A Software Engineer working to a tightly drawn specification is unlikely to bring as much to a project as one whose opinions and ideas for enhancing the piece are warmly welcomed by the artist. On a more cautionary note, all parties must take steps to avoid over-burdening the project with technological goodies just because "anything is possible".

Embodying the Artist's Ideas

Regardless of the model for collaboration that is in force, the simplest approach to providing the software for a digital artwork would be to ask the artist to specify the sequence that is to be followed by the piece. The Software Engineer will then go ahead and encode the specified sequence directly within the software. For simplicity, let us consider an open-loop artwork that has a series of LEDs that can be lit independently to produce some effect or other. Early collaboration with Esther Rolinson at one of Loughborough University's COSTART residency sessions [5] concentrated on this

kind of work and the artist in question was keen to learn about the possibilities of such a scheme. Let us further establish, for this argument, a simple "sequence" that illuminates a single random LED for a random, although bounded, time and to repeat this forever. The appeal to the viewer of this particular piece is likely to be very limited. Maybe the only appeal to the viewer would be in attempting to predict the next change. Such behaviour would be easy to permanently embed within the software that controls the artwork and indeed this approach would be completely justified. We must, though, assume that the artist has something more demanding in mind and wishes the artwork to portray some artistic merit, beyond, of course, its physical structure.

It is at this point that we must look to the artist to provide some form of specification that will embody the artist's ideas within the LED sequence. The artist, almost certainly working independently, will sketch out the proposed sequence and pass it to the Software Engineer, who will then encode the sequence into the software that is to control the artwork. The artist will doubtless be astounded by the ease with which the artwork follows the intended sequences, but will equally likely become bored with the repetitiveness of those very same sequences. The only way forward is for the artist to go back to the Software Engineer and ask that the sequences be amended or expanded. This process could repeat itself many time over before the artist is sufficiently satisfied to allow the artwork to be seen in public or until the patience and resolve of the artist and Software Engineer finally expire. Even when this cycle comes to an end, it is probable that the artist will be frustrated with the artwork and, having been seduced by the possibilities of the technology, will be hankering after bigger and better things. Further enhancements are most unlikely to see the light of day, due either to a reluctance to bother the Software Engineer or exhaustion of the budget. Experimentation with new ideas will be stifled by the same limitations.

Of course, there is very little difference between this scenario and that faced by businesses that took the innovative and largely untried step of utilising computer systems many years ago. In the early days, before the advent of *shrink-wrapped* software, two routes were open. Companies either employed their own programmers or enlisted the help of software houses (often the computer hardware manufacturers) to write systems to support their business processes. In either case, they too had to specify the operation of the software beforehand and changes to the software's operation that were desired, or simply became necessary, had to be implemented by the same teams. Computerising even the simplest of business processes required this sequence of events.

Increasing Flexibility

We are now at the stage where the appetite of the artist has been whetted by the successes of a simple system and there is a desire to produce an enhanced design. Clearly, the artist could re-engage with the Software Engineer and build a specification for the improved version. Learning from earlier lessons, though, it becomes apparent that the task of accurately specifying a largely unknown system is tricky. The unknowns relate to both the artwork - maybe its structure and certainly its requirements - and still, in spite of earlier successes, the capabilities of the technology.

In solving the obvious dilemma, we turn again to the history and practice of business computing. Nowadays, many of the less demanding business processes are implemented using spreadsheets or simple database systems rather than relying upon bespoke programs. In this situation, the spreadsheet program has been written to allow the user to configure a spreadsheet to solve a particular business problem. When the user wishes to add an extra column of data, there is no need to consult the team who programmed the spreadsheet application. Instead, the user simply points and clicks and immediately the new column is added. Its functionality is then easy to define, perhaps with the insertion of a formula linking cells in other columns. In this situation, we are merely defining some kind of function that maps certain inputs to certain outputs and we thus find ourselves in essentially the same domain as digital artwork control. This clearly begs the question as to whether the creators of digital artworks should be required to behave as the clients of software houses did in "the dark ages" of the 1970s. Should they instead expect a more modern approach?

Artists from many domains, including installation artists, are familiar with software packages such as Macromedia Director, which allow them to draw and animate scenes. This kind of software provides the artist with a powerful tool, which assists in visualising the piece even at the very early stages of its design. Being aware of this kind of software adds further to the artist's frustration in being unable to so quickly and easily convey artistic desires to the programmer. Products such as Director, Flash and Max encourage the artist to experiment with new ideas, providing them with the powerful editing and animation tools required for the job. The artist is set free from having to specify the requirements of the piece beforehand. It is clear that this kind of approach must be our goal for capturing the artistic intentions of the installation artist making use of computer technology.

Proceedings of the 20th Eurographics UK Conference (EGUK'02)
0-7695-1518-5/02 $17.00 © 2002 IEEE

It is only by allowing flexibility in the programming of the piece that the creative processes can be fully enabled. At odds with this approach is the fact that the microprocessor technology that is often at the heart of the artwork's controller is not particularly amenable to supporting large point-and-click applications of the spreadsheet or Macromedia Director genre. In some situations, the solution to this problem is to encourage the artist to make use of established, simple programming techniques. Many artists are familiar with the ideas behind programming and will be quite happy using a programming language such as *Basic* to produce the kind of programs required to perform the necessary control. However, it may not be possible, for reasons of scale, to embed a *Basic* interpreter into the artwork's controller or to locate a compiler that produces the control processor's native code. It should be understood that the type of processor used in the controller could have as little as 8Kb of ROM and 256 bytes of RAM. What is required in this situation is a simple application-specific language, which can easily be interpreted in real time within the microprocessor domain.

In this scheme, the Software Engineer expends no effort on encoding sequences, but instead puts time into providing the mechanism already alluded to. It has to be said that the time taken to achieve this sequence interpreter is far in excess of the time that it would take to encode sequences directly, but the benefit to the artist is out of all proportion to the extra effort. The level at which this works can be illustrated by returning to our simple example of the set of LEDs. The intent of the following fragment from such a sequence is clear.

```
REPEAT  20
ON      6, 7, 8, 10, 12
PAUSE   10
OFF     ALL
ON      3, 4, 9, 14
PAUSE   10
LOOP
```

It causes the five LEDs nominated in the first `ON` statement to be illuminated for 10 seconds and then for that condition to be overridden by the illumination of the four LEDs nominated in the second `ON` statement for a further 10 seconds. The whole sequence then repeats twenty times before moving on to something else. Further instruction types, not involved in the fragment shown, offer shift and rotate operations, conditional and unconditional jumps and subroutine calls.

The interpreter for such a simple application-specific language is straightforward to write in even the simplest of order codes found on the smallest of microprocessors. A substantial added benefit to the Software Engineer is that the resulting system stands much more chance of being re-usable in a future project. It is probable that, at most, extensions will have to be made to the application-specific language in order to accommodate new requirements. For example, should a future project require analog control of lighting, the necessary instructions can easily be added. The available budget will almost certainly allow the Software Engineer to improve and enhance the interpreter each time it is utilised in a new project.

It is more than likely that the controller that is integrated into the artwork will not have a keyboard and screen present and so these sequences will have to be developed on another computer - a PC or a Macintosh - and then uploaded to the controller. Straightforwardly, the sequences can be presented in textual form to this *host* computer, where they are checked for syntactic accuracy. Subsequently, they are translated into an intermediate tokenised form, which is more economical than the textual form for both transmission to and, perhaps more importantly, storage within the controller. Clearly this process also avoids the need for the controller to perform the translation step. The resulting whole can easily be accommodated in a small package, both physically and from the point of view of cost. In terms of the evolution of computing in the business environment, to which we referred earlier, the present situation is much like that at the time when SQL (Structured Query Language) was developed for manipulating databases. SQL is essentially an application-specific language, which allows records to be stored in and retrieved from a database that has already been set up. Of course, SQL is a long way from the point-and-click nature of today's business and indeed consumer applications.

The resulting scheme should be more than acceptable to the artist and presents, in the author's view, an easier, more focused interface than that provided by established programming languages. However, we are still some way from the point-and-click environment of a product like Macromedia Director, which must still remain our ultimate aim.

Visualisation Through Simulation

We will need to assume for the moment that the artist will be able to master the idea of writing these sequences in order that we can examine what else the host computer can do to assist the artist. One of the problems that artists face is that before the artwork controller is installed, it is difficult, if not impossible, to visualise the effect of the sequences that have been programmed. With that in mind, it would be beneficial to provide the artist with some kind of simulator for the

piece. It is straightforward to produce a simple on-screen graphical representation, in two dimensions, of the artwork and the structure into which it is to be embedded together with a representation of the elements that are to be controlled, such as the LEDs in our example. The simulator can "run" the sequence and portray the results on the graphic. However, we can do much more.

As part of the process of bidding for a commission, many artists produce very faithful three-dimensional representations of their proposed work. Where this is done using a Computer-Aided Design (CAD) or other drawing package, we can capitalise on this work. The image produced by the artist can be captured into the simulator and the controlled elements can be projected on to this image. In the case of a public artwork, it is common for artists to build these representations upon photographs of the surrounding area. Figures 1 and 2 show images produced by Esther Rolinson, following precisely that process. What we see here are merely 3-D CAD drawings of the piece superimposed upon

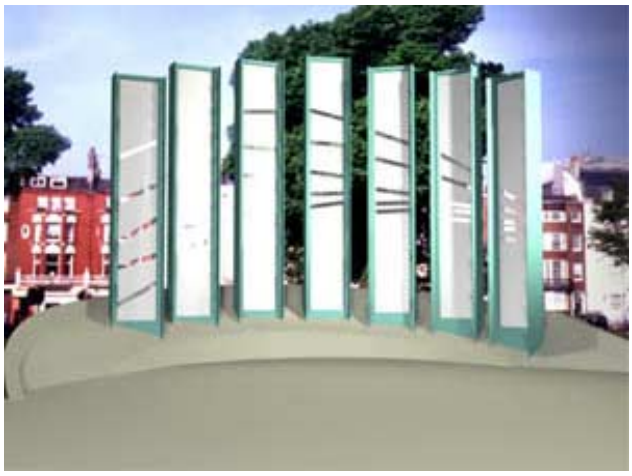

Figure 1 Artist's impression of the piece in situ, in daylight

photographs of the proposed site. It is worth noting that the artist has proved, in this exercise, great ability in manipulating images using complex software products. The artist's obvious familiarity and comfort with the technology encourages us to find new ways of allowing the expression of artistic input to the specification of the control process.

It is useful at this point to describe the piece in order that the appearance and operation of the simulator can be more fully appreciated. The piece in question consists of seven steel columns, each containing a sandwich of acid-etched glass, with certain areas left transparent, and panels of a material known as Priva-Lite® [6]. The Priva-Lite® panels are positioned so as to coincide with the transparent areas of the etched glass panels. Each

Figure 2 Artist's impression of the piece in situ, at dusk

Priva-Lite® panel is itself constructed as a sandwich of glass and a material that is opaque until an electrical current is applied, at which point it becomes transparent. These panels could be equated to very large segments of a liquid crystal display (LCD) and indeed make use of the same fundamental technology. In this context the panels behave very much as window shutters in a building. In this piece, five Priva-Lite® panels are used to obscure the clear panels on each of the seven columns, with the panels being individually addressable by the control system. A significant part of the artistic input to the piece concerns the positioning of the clear areas within each column and the simulator can help here too.

Figure 3 shows the simulator [7] built for this project. What we see in the upper right-hand window of the simulator's space is a two-dimensional representation of the seven columns with the five Priva-Lite® panels superimposed on their respective surfaces as lighter areas. The simulator allows the artist to draw the panels on to the two-dimensional surface, offering a significant insight into the final appearance of the piece. Further, the simulator is not limited to seven columns with five panels on each. Rather, configuration functions allow the artist to choose the numbers of each, their size and their spacing. Below the 2-D window is a projection of

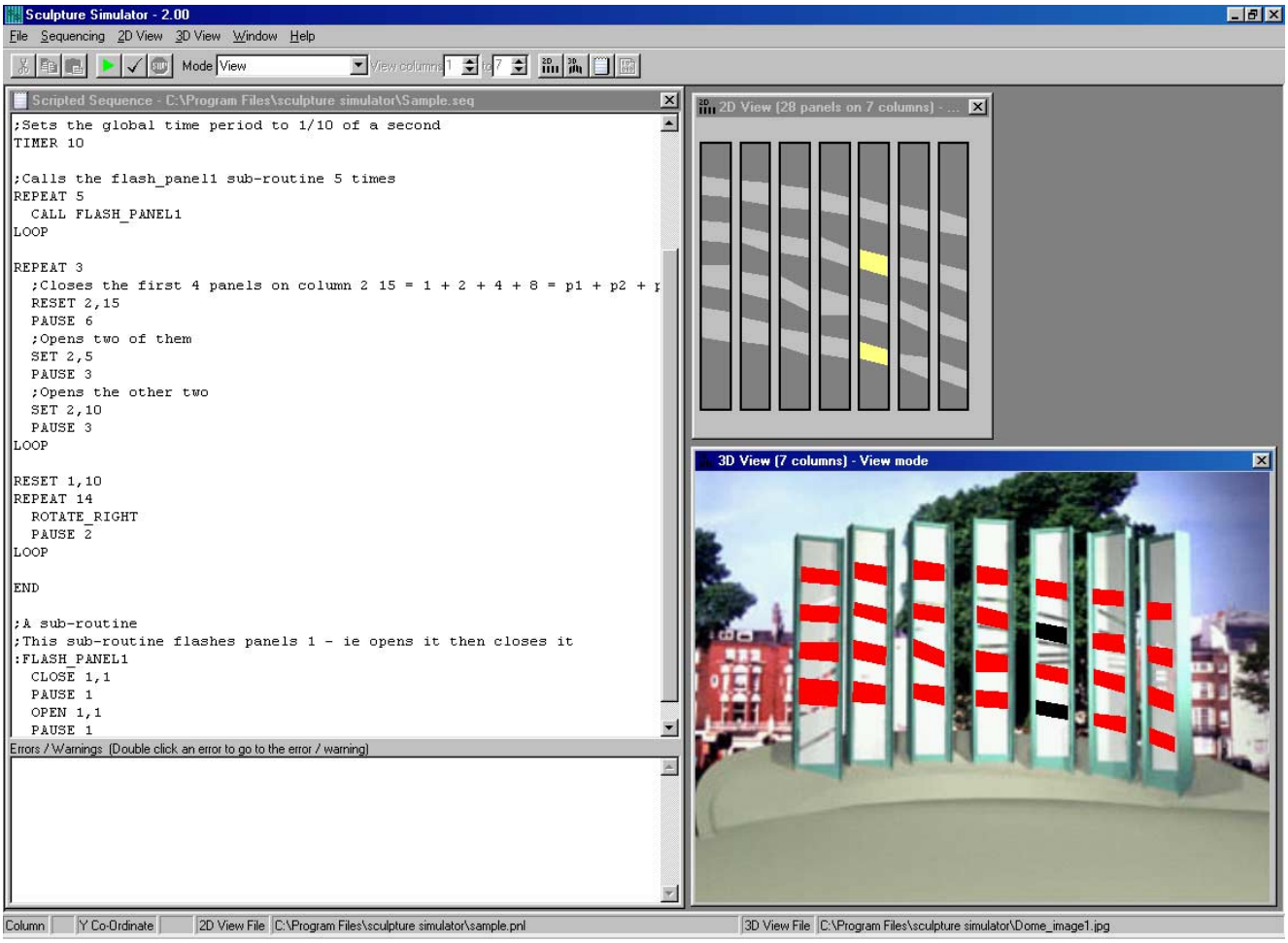

Figure 3 The Simulator

the seven columns, with their Priva-Lite® panels, on to the 3-D artist's impression of the artwork in situ, taken from the image that we have already seen in figure 1. Again, the simulator allows careful positioning of the panels on the 3-D representation. The colours used by the simulator to portray open and closed shutters are arbitrarily chosen by the user. The selection can be made by the artist to allow the simulator to provide a realist representation even with a background such as that shown in figure 2, where the piece is seen at dusk.

A sequence to operate the artwork is shown in the window to the left and inspection will reveal that the instructions used to construct the sequence are very similar to those already specified in our simple LED example. `SET` and `RESET` instructions render panels opaque and clear, respectively, with each instruction using two parameters to identify target columns and panels. The simulator provides the syntax checker and the translator from the text-based representation of the sequences into the intermediate form referred to earlier. It then proceeds to "run" the sequence exactly as the real control system would do. As the sequence runs, the changes in the Priva-Lite® panels are reflected in the 2-D and 3-D views simultaneously, by changing the colours of the panels. This whole process restores a degree of immediacy to the artist's actions. Changes in the sequence can immediately be seen in the simulator. Painters and sculptors have the benefit of seeing the results of their actions as soon as they occur. The simulator goes some way, figuratively at least, to putting the brush or the chisel back into the hands of the digital installation artist. Although we are still some way from our point-and-click goal, we have taken a major step forward in allowing the artist to visualise the work and the manner in which it can be controlled.

Edmonds and Quantrill [8] suggest that technology has an impact on artists in three distinct areas, namely in *thinking*, in *making* and in *communications within the team*. Further, as has already been alluded to, there is a likely impact on artists and technologists alike of aiding in the *development of the technology*. Artists will sometimes come up with an idea that is outside the scope

of the technologist's current thinking and will effectively be "pushing the boundaries" of the technology. In coming up with a solution, the technologist will have to advance the field (or at least the local understanding of that field) and in doing so might make discoveries that will further enhance the artist's view of what is possible. There is clear potential here for each party to feed from the other's experiences and each is pulling along the other. Providing the simulator clearly makes a contribution to the making phase, as it allows the artist to visualise the sequences that are to be used in the artwork. It could also be said to be making a contribution to the communication aspect, because it allows the artist to understand the possibilities of the piece or of future designs. This useful tool could be used in this or a different commission, to actually assist with the design and will clearly contribute to the thinking phase.

We have already noted that the simulator allows the numbers of columns, their spacing and the numbers and positions of the panels to be changed. This simulator was originally designed with the particular artwork in mind, but it would not be difficult to imagine a situation where the artwork's design (the creative bit) and its operation (the technical bit) are integrated into one package. All that we would need to do is to add a 3-D drawing function to the simulator. Conceptually, this is straightforward, but in practice would be very time-consuming. Existing 3-D drawing packages have resulted from many years of development along with a great deal of financial investment and so it is not really feasible to contemplate this step here. However, we have seen that the simulator will accept images imported from other drawing packages and this is considered to be sufficient at this stage.

Once the basic simulator has been constructed, we can add further functionality. Merely as a demonstration of what could be achieved, the present simulator contains a rudimentary *bill of materials* function. It calculates the respective surface areas of glass and Priva-Lite® required for the project and estimates the cost. Although a relatively simple thing to achieve, this feature is useful, given that the cost of Priva-Lite® is extremely high, at almost £1800 per m². The artist can have an immediate indication of the impact of certain decisions on the total cost of the project.

The Next Step

So far in this argument we have been assuming that the artist is comfortable with sequence programming and that it is appropriate to the piece. However, what if it is deemed to be inappropriate? What is needed is a point-and-click interface to the programming of sequences. Some of what is necessary to achieve this step is difficult, though, especially if we are looking for integration with the design process. In any event, a major requirement would be the need to integrate a user interface that can capture the artist's creative processes and enable the identification of the elements to be controlled, be they LEDs, Priva-Lite® panels or whatever else. We would then need to have the software be able to visualise those elements on the screen.

All of this is conceptually achievable and is indeed only a relatively small step from the existing simulator. It remains only to enable the specification of different kinds of controlled elements. There is a parallel here in the simulation of electronic circuits. Standards, such as SPICE (Simulation Program with Integrated Circuit Emphasis) [9], developed at the University of California, Berkley, exist for the specification of the behaviour of electronic components. Mathematical models of each component are processed in sequences and in a manner defined by the circuit under simulation. So, the technology to achieve this goal exists in other related spheres and would adapt relatively easily to this application.

Perhaps the most difficult aspect of this step is the manner in which the required behaviour of the artwork is to be captured. Attention has already been given to capturing the artwork's static conditions, albeit that we have noted that integrating this into a single application would be non-trivial and expensive. Of much more difficulty, though, is capturing temporal information defining the timing of the piece. Capturing any form of temporal information appears to pose a problem, then. However, in some areas of digital art capturing temporal information is inherent. In music composition, for example, the composer can enter each part of the score in real-time using a piano-style keyboard. Amateur software exists for capturing music from MIDI-enabled keyboards and its ready availability in the freeware and shareware domains suggests that the process is reasonably straightforward.

Returning to our requirement, though, the system could enter into a dialogue with the artist, but this would serve only to restrict the flow of the creative juices. However, capturing timing information on the flow of a complex piece, such as we have been examining, is more difficult. We cannot simply use a point-and-click scheme to define the operation of the piece as it takes time to move the mouse pointer from one panel to another and this time distorts the "real-time" elements that we strive to capture. How, then, can we represent simultaneous changes in two panels? Should the aim be to be able to develop a system that provides an interface more akin to brush and canvas? Work has been carried out in

capturing gestures in multi-media systems [10] and using these gestures to enhance the system's "understanding" of the user's intent. This work, with its emphasis on touch-sensitive screens, could be embraced in order to assist with the current challenges.

Whatever the route to the solution, the problem increases in complexity substantially as soon as we contemplate artworks that respond to their environment. How can we provide a system that captures the artist's view of what should happen when a particular stimulus is applied? If an artist wishes an array of LEDs embedded in a footpath to emulate the swirling of the water in a puddle as the system senses the wind blowing across it, how could this be represented? A naïve approach would be to say that all we would need to be able to do is to describe the relationship of the inputs to the outputs and this has indeed been done, at least as a mathematical model. However, describing such a relationship in this context, given the required artistic input, would be by no means trivial.

It will be necessary to work with artists in order to determine what they desire in such a product and to attempt to identify new ways of capturing the information that is required. Active drawing surfaces have been aiding artists for some time, although there is debate about their effectiveness and whether or not even this advanced technology actually inhibits the creative process [11]. More work will be required to determine the manner in which various technologies might be integrated in order to be able to capture the creative aspects of a digital installation artwork.

Conclusions

When working with artists, Software Engineers have a challenge to bring the technology, which is very familiar to them, within reach of the artist. As the complexity of artists' work increases, the level of programming that is required also increases significantly. As soon as we consider installation artworks that respond to their environment, the processes involved in specifying the response of the piece become much more difficult. Some steps have been taken towards the goal of providing artists with an interface to the specification stage that does not inhibit their creative talents, but there is some way to go. We have seen how point-and-click software can assist the artist is visualising the artwork and that this can lead to the artist being assisted in the concept and design process too. What is required now is to carry out further work in order to establish methods that can be utilised in future requirements capture software. By working with artists in the installation art field, we can seek out ways to enable the capture of the artist's ideas without inhibiting the artistic process.

REFERENCES

1. Tate Britain. Turner Prize 2001 - Martin Creed's Work, http://www.tate.org.uk/britain/exhibitions/turnerprize/Creedwork.htm.
2. Rolinson E. Shifting Spaces, Explorations in Art and Technology, Springer-Verlag, Candy L, Edmonds EA (eds.), pp 227-232, 2002.
3. Machin CHC. Realising Digital Artworks, Explorations in Art and Technology, Candy L, Edmonds EA (eds.), Springer-Verlag, pp 135-142, 2002.
4. Candy L, Edmonds EA. Modelling Creative Collaboration, 5th International Roundtable Conference on Computational and Cognitive Models of Creative Design, Heron Island, Australia, 2001.
5. COSTART Project - Artists-in-Residence, Loughborough University, England, http://creative.lboro.ac.uk/costart.
6. Priva-Lite®. Product Information, St Gobain Glass, http://www.sggprivalite.com/en/intro/intro_fr.html.
7. Garton DE. Design and Implementation of a Software System to Aid Esther Rolinson in Designing the Brighton Dome Sculpture, Final Year Undergraduate Project Report, Department of Computer Science, Loughborough University, England, 2001.
8. Edmonds EA, Quantrill MP, 'An Approach to Creativity as Process', Reframing Consciousness, Ascott, R. (ed), Intellect Books, Second International CAiiA Research Conference, UWCN, Wales, August 1998, pp 257-261.
9. SPICE (Simulation Program with Integrated Circuit Emphasis), University of California, Berkley, http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/.
10. McKenzie Mills KC, Alty, JL. Investigating the Role of Redundancy in Multimodal Input Systems, Gesture and Sign Language in Human-Computer Interaction, Proceedings of the Gesture Workshop 1997, Wachsmuth, I. and Frohlich, M. (eds.), Springer Verlag, Proceedings of Gesture Workshop 1997, Bielefeld, Germany, 1997, pp 159-171.
11. Quantrill MP. On the Creative Use of New Technology by Artists, Creativity and Cognition Research Studios, Department of Computer Science, Loughborough University, England.