12 CASE STUDY

Programming for design: Rob Saunders

Rob Saunders is a programmer and works as a freelance consultant in the media industry with a range of clients. He is also employed on non-commercial projects by people from other areas such as mathematicians, scientists and artists.

With a background in computer science, Rob is a specialist who has considerable expertise in the programming that goes into artificial intelligence and artificial life systems. The work he does often includes developing highly complex programs that use this level of programming knowledge to generate intelligent artworks and complex design systems. In particular, this leads Rob to work as a consultant with design firms where his programming skills contribute to a project alongside the expertise of other new media practitioners and design specialists to create innovative web-based projects. Programming with the sophistication and the depth that he is able to operate at can enable a new form of creativity and visual outcome.

In this interview, Rob discusses how he collaborates with other members of a creative team and develops a project. He also addresses how, as a specialist working with programming and data management at a deep level, he is at a different level from that of his partners and how he manages his contribution to a larger project without allowing what he does to dominate the design or creative concepts.

Background

As a programmer, you are closely involved in creative practices. Did this come out of your background and education? What was the pathway through which you got involved in this particular approach to new technologies?

I guess I've always been involved in new technologies. When I was twelve, my parents bought me my first computer; it was a gorgeous little machine called an Oric-1 with a massive 16 k of RAM. Unfortunately, hardly anyone else ever bought one, so in order to play games I had to learn how to program them myself. Pretty soon I found that programming was more fun than playing games.

At school I was always good at science, maths and art; it was obvious that I was going to have to make a choice between studying art or science at university. It was around this time that I decided that what I really wanted to do after university was to develop computational tools for creative people. I had experimented with using my

computer, an Atari ST, in my artworks for A-level, and was excited by the creative use of computers by people like John Lasseter who went on to set up Pixar.

Moving on to higher education, I chose to study Artificial Intelligence at Edinburgh University which I saw as the most exciting way for me to continue studying computers. While at Edinburgh I became fascinated by the work of William Latham and Stephen Todd at IBM, using computers to interactively evolve 'virtual sculptures'. After finishing my degree, I decided that I wanted to continue studying evolutionary art and design systems to a doctoral level. Eventually, I became fascinated with the nature of exploration in design and this led me to develop a computational model of curiosity to help me investigate curiosity as a motivating force behind the exploration of design possibilities. The study of curiosity in art and design remains the main topic of my personal research.

What are the areas that you are now particularly interested in working in and that you think have potential for the junction of creativity and high-level programming that you are involved in?

I plan to continue working with artists and designers and hopefully expand to include collaborations with musicians. I'm also very keen to start exploring ways to create intelligent design tools for people who don't think of themselves as designers, e.g. hobbyists.

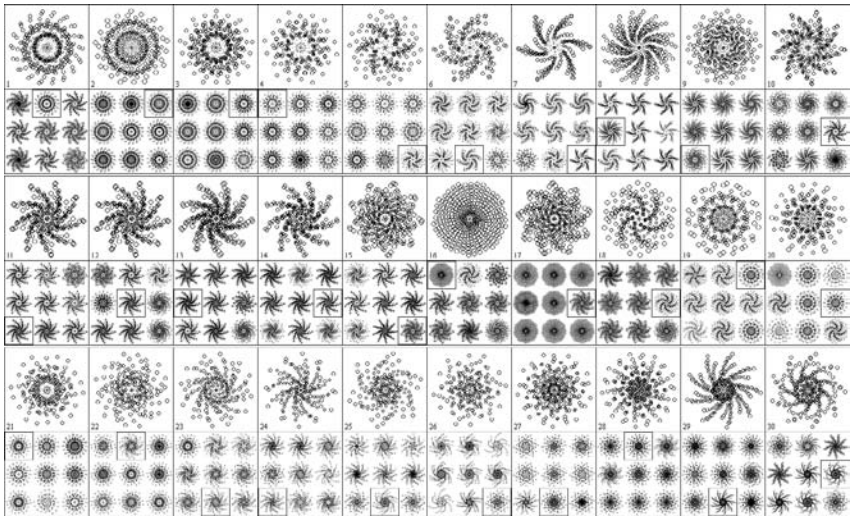

Figure 12.1 Rob Saunders, illustration of the way a curious design agent autonomously explores a space of possible designs (2001)
'A curious design agent was given a simple design tool to allow it to explore the space of possible designs. From top-left to bottom-right the illustration shows a short sequence of the designs that the agent produced. An important thing to note about a sequence produced this way is that the agent moves the exploration process forward by selecting similar-yet-different designs from those that it has seen before. This novelty-seeking behaviour is similar to that observed in human designers as they explore design possibilities during the initial phases of design' (Rob Saunders).

Personally, I continue to research motivations for the creative process including curiosity. I continue to build computational models of curious design systems exploring various design domains.

Collaborations

When you are working on a professional project, what is the basis of the relationship that you have with your client? Has the company usually defined the nature of the project and recognised that they require a level of programming and particular specialism to achieve it, or are they more likely to recognise that you can provide an extra dimension to a project but leave what it might result in up to you?

In general, I'm brought into a project when a team has a good idea of what they want but no idea how to do it. Typically, this means that I'm provided with a brief that lacks a lot of detail and the first thing I have to do is work out what can be achieved in the time available and then start to produce technical demonstrations.

The designers I work with are also skilled programmers, so when they need me to get involved it is because they need something beyond simple computer programming. Typically this will mean that the project requires the development of an intelligent system.

Is there a typical relationship and delineation of skills or is every job or workplace situation different?

I have found that each job is different. In general, I like to delineate my job so that I do not unduly influence artistic or design decisions, but sometimes that isn't possible. Sometimes, there is a natural synergy between the software I develop and the artefacts that are produced, but sometimes the influence that the software I produce has on the creative process is accidental, at least it is so on my part.

Defining 'intelligence'

Terms like 'intelligent system', 'intelligent agents' and 'artificial intelligence' are used broadly and variously in different parts of the media industries, the artistic and research communities, not to mention computer science, where the meanings may be very specific. How are they understood in the professional contexts you work in, such as when you talk about an 'intelligent graphic design system'?

The phrase 'intelligent system' is just shorthand for a system that uses artificial intelligence technology. It is a convenient shorthand for people who work in this field although it must be noted that the term should not be taken too seriously, very few intelligent systems are really intelligent. This is one of the problems that the artificial intelligence community has in general, by choosing such an evocative name the field doomed itself to failure by virtue of people's expectations of what 'intelligent' means.

Intelligent agents are software agents that use artificial intelligence technology. Agents are pieces of software that have some autonomy, in contrast to applications and objects that always do as they are commanded to. I simply use the term 'intelligent graphic design system' to indicate that a system uses artificial intelligence technology to accomplish graphic design tasks.

Using technical demonstration models and prototypes

Projects that operate at the level that you are describing often require the production of a technical demonstration model or a prototype, whereas less complex new media projects developed in the same context, such as product branding, may just have in-house testing before being released. How would you describe the need for a technical demonstration and a prototype and what would you describe as the difference between them?

From my perspective they take the same form but serve different purposes. A technical demonstration shows whether something is technically possible or not. An example could be devising a program for a network layout problem, which might show that a crucial problem could be solved. Another example would be when working with artists I might choose to present a demo that illustrates the problem of something fundamental to the system, in which case it might not try to solve any problems, it simply demonstrates a problem.

One of the reasons that I find technical demonstrations to be useful is that they allow me to show the limits of what is possible in a positive way. I find this to be particularly important because it is often difficult for people who want to use artificial intelligence to separate fact from fiction about what is possible.

In contrast, a prototype is something that attempts to be a partial solution. For example, sometimes I work with artists who are working on projects that use quite rich and complicated data, and my goal is to create a visual model. I might first ask for a specification and from this I would create a first version of the visualisation.

However, my experience working on several projects with artists has made me wary of the disadvantages of developing prototypes, as they can have the unintended effect of setting in stone arbitrary design decisions that I have made whilst programming. Consequently, I try to limit how much effort I put into the look of prototypes so that collaborators don't get too hooked on a particular visual style.

Visualising data and facilitating designs

Visualising data is obviously very important in your field and for many new media practitioners, including artists, the challenge is to find ways to make information that is otherwise to complex to handle, comprehensible and manageable. When you are brought into a collaboration to work in this area how do you go about negotiating what can and can't be achieved in terms of the visual output?

In some cases, I have been contracted to solve a geometric problem, such as in the case of a project to develop software to intelligently lay out complex network diagrams. In cases like these, it is natural that the software I write has a significant impact on the look and feel of the user interface. Ideally, I prefer to create systems that can be easily customised through data files. By using a data-driven approach to the development of creative support software, many artistic and design decisions can be changed without the need for additional involvement on my part.

This was the approach I was recently able to take with the development of an intelligent graphics design system, where I provided a number of different ways that the performance of the system could be changed by editing simple data files. In this case, my client was a small start-up company that saw an opportunity for bringing a new

graphical tool to an industry that has to deal with complex data about large networks on a daily basis. Their first product required the founders of the company to produce complex graphics by hand that were then used through a simple user interface.

When the decision was made to produce the second version of their product, the client decided to approach a design firm about redesigning the interface and explore the possibility of generating the graphics automatically. The client brought in a design firm, who were able to determine the initial requirements for the new user interface but they did not know how to specify the intelligent graphic design system.

At this point I was set the challenge of developing an intelligent system that could lay out the complex data in a graphical form. At first this was just an informal challenge set by the lead designer, who is a good friend of mine. Over a couple of free days, I was able to come up with a proof-of-concept that made it clear that the problem could be solved to everyone's satisfaction.

I was then contracted to develop a more complete solution that has been shipped with the second version of the system. Although the original proof-of-concept only took a couple of days to develop, the production version took almost nine months of working one-to-two days a week to complete. The result has been a great success and has helped the company to continue to grow its market.

I am currently working on the second generation of the intelligent design system that will be integrated in the next version of the system, due to be shipped in the second half of this year. Instead of being contracted, however, I have negotiated a share in the company in return for my contribution to the business.

Working with data files

When you are describing 'data files' do you find that different people have a different understanding of what 'data' is even though all interaction with software is based on the management of data? For many practitioners, the dominant metaphor for the way that we work with data is the database, in other words that data is packaged in a modular way so that it can easily be amended, sorted and searched. The experience for most users is one of top-down management of the data using sub-sets or fields and the complexity of data at the bottom level is rarely engaged with. Does this metaphor translate into the programming work that you do?

In general, this understanding of what a program does in terms of processing data is correct, however, not all data is of the same type.

The distinction between different types of data files can be illustrated by considering the difference between a Word template and a Word document. Essentially these are very similar types of data files (almost identical in fact), but a Word template can change how the program works when editing another data file, i.e. a Word document.

So when someone working in my context talks about controlling the behaviour of a program through data files we are talking about data files that fundamentally change what a program is able to do, rather than simply changing the 'raw material' that the program works with. In the case of one project where I created intelligent design agents to generate company logos, I had designers create files that specified the basic shapes, filters and operators that the agents were able to use. Consequently, these data files defined the space of possible logos that the agents were able to create.

The distinction between program and data files can be further blurred when one looks at things like macros commonly used in templates, etc. These are small programs that

are written in a language designed to be easier to use than the one used to write the main application so that 'casual programmers' can make more significant changes to the behaviour of a program. Many applications now support some form of embedded programming language.

When you are working on highly complex projects do you find that your partners also need to be very experienced or adept at this way of working to collaborate effectively? Do they need to have a very good sense of what the program could achieve at its deeper levels through the management of data files?

No, this is not necessary. It is often possible to develop computational systems in such a way that a significant amount of the system's behaviour can be controlled through the use of data files without the need for a detailed understanding of the program's inner workings.

Many systems that require collaboration between programmers, artists and designers are developed this way. Good examples can be found in the development of video games. Modern video games often take years to develop and require dozens of people with different backgrounds and skills, so a data-driven approach has become the standard way that such systems are developed.

Rob Saunders' website is www.robsaunders.net