 [1] **Andrew Bluff: Digital Artist Researcher**

The software application that comes out of this coding, does act like a creative partner in an artwork.

In Andrew Bluff's creative practice, research, art and design are distinct but complementary elements of his reflective practice in which computer programming plays a central role[2]. He created the mobile apps *DrumStudio* and *RoboDrummer* and received the App Art Award for *Mobile Phone Orchestra.* He has collaborated with people from diverse backgrounds for several years during which time he completed a practice-based PhD on 3D techniques for the augmentation of live performance[3]. His art involves designing interactive software for live performances that can transform the traditional practices in drama and dance in collaboration with Stalker Theatre[4]. He combines his creative work with ongoing research at the Animal Logic Academy [5].

Creating artistic digital forms has become a way of life that transcends the need for solo activity and ownership. Co-creation offers more sources of inspiration and access to unusual perspectives that only working with people from diverse disciplines can bring. His satisfaction lies in knowing the value of a contribution that is in one sense concealed, but in another, is very apparent. As an artist and a researcher, he distinguishes between reflection in design, art and research. Reflecting through research is integral to exploring new ground for creating art. In the art making, he reflects on feedback from participants who experience the work but when designing software, reflecting involves reframing problems during the programing process. Using digital tools can have reciprocal effects on the way practitioners think and act creatively, but as a computer programmer he can exercise control over the whole process. For Andrew, creating new digital systems offers more opportunity to make innovative artwork and that is where his strengths lie. Bringing his own thinking style together with coding skill is, he believes, essential to creating complex artistic systems. At the same time, as he observes, it is a two-way street: "*you also shape the program you are making to adhere to your own unique way of thinking*". The software he creates to suit his needs becomes a collaborator in making a work, and this imbues the relationship with a sense of partnership.

Figure: Entangled in Stalker's *Creature Interactions*. Photo by Andrew Bluff.

An interview with Andrew follows in which he describes his practice and expands upon the part played by digital technology in augmenting his practice in such a way that he is able to augment the practice of others.

Interview

Q: How do you work creatively?

A: Over the past few years, I have worked with Stalker Theatre, on interactive particle-based visuals that react to the movements of the performers. David Clarkson, the director, places emphasis on really getting to know his team. He chooses wisely and has a collective of artists who have worked on nearly all of the productions. We tried to develop the show with all artists in the same room where possible. I made sure I was in the space the whole time. But saying that, I was in the space working during the day but on my train journeys to and from the central coast, I would do a couple of hours work and maybe a couple of hours each night at home as well, so I put in three or four hours of solo work per day on top of the integrated development time.

Q: Is collaboration beneficial to the way you work creatively?

A: There are pluses and minuses. The pluses are you get inspiration and other perspectives that change the way you might think and look at something. It also lets you off-load some parts of artistry that you don't enjoy- which for me is called 'marketing'! Also putting teams together: I like it that Stalker has physical performers and musicians that I wouldn't necessarily thought of working with. You meet people you don't expect to meet and that's really enriching. That's what I like about the university as well; you keep bumping into interesting people, researching random things, it gives you different perspectives. I have found in my collaborative work that I have not, to date, and I probably never will be, the key driving force. In that regard, since I've been doing collaborative work, I haven't created my own true artistic piece. I made that one mobile phone orchestra which was all my own doing, and that was quite successful. The difference is when I am in a collaboration I reflect and react to the other artists' work in producing what I produce. It's iterative and I dare say they reflect on what I produce and vice versa which is a very fruitful way of working. But there's not that core searching for some notion of the world that doesn't make sense to you, and really diving into that, getting involved in that personal attachment and then having the work drive out of that. I can't really imagine, at this stage, doing art and not collaborating. I think it takes an enormous amount of strength and will to be doing independent art. In my life at the moment I am happy and comfortable to be doing the collaborative work so I am not seeking that.

Q: What does the concept of reflective practice mean to you?

A: Reflective practice to me describes a practice which exists through a cyclical process of action and reflection. Do something (action) and then assess (reflect) the resulting pros and cons of this action in order to refine or redirect the next course of action. My art involves designing bespoke interactive software that I can use in live performances, and my research is around how this technology can alter the traditional practice of live performance. In my own work, this cycle occurs in different but interrelated scopes depending on my current role of artist, designer or researcher. As a designer, I use reflective practice to evaluate and find the optimum solution to a given problem. As an artist, I use reflective practice to find new problems to explore. These new problems result from reflecting on previous artworks and the design process of creating technology. As a researcher, I use reflective practice to identify frameworks and phenomenon across the body of artistic work, both mine and others, and use this framework to then find unexplored areas with which to focus new artistic works or research.

As the scope of reflective practice expands, the scope and formality of reflection and analysis expands. My reflection on design is very internal and solo, reframing problems in my own head or on a piece of paper, while my artistic reflection will often incorporate the criticism and suggestions of others via informal chats. The research will involve conducting and recording interviews with fellow artists and performers and applying formalised research methods to generate frameworks and identify phenomena in the hybridised artistic practices. I am not suggesting that this tiered notion of reflection and analysis is in any way universal, but it seems to apply to my particular brand of software design, interactive art and practice based research. I guess I'm doing an 'instant in-action reflection'…I like to make each parameter of the module I've just made easy to manipulate in real-time. Just looking at the response you're getting from the program especially when you are pushing it to the edge of its limits. I guess you are reflecting on the artistic potential of the object you've just made, judging it instantly as you are doing it- in-action. I'll do that for a period of time and then I'll sit there and reflect on what I felt worked and what didn't work. I do a lot of walking around the room. I 'm staring off into the distance, walking around the space and that's my reflecting on the programming side of things more than anything.

After I've made something artistic and I've played with it, my reflection is more going and having a break- watching some terrible television or something. And then my mind will wander back onto it. I think there are different stages. There probably is that same day and you are thinking about it a lot so when you are making spaghetti or something, it will pop into your head and you're re-assessing it. But then roughly two weeks later you find that you're thinking about it again, maybe in a different way, maybe in line with seeing some other artwork, seeing someone do something else and it will make you see some connection between what you are doing and you change it. If you are working with someone else you might be working towards a milestone or towards a development period or a show. After that you have a break of a week or two.

Q: What is the main satisfaction coming out of the kind of work you do?

A: It's in the making of the work. Just being interested in what I'm doing on a day to day basis is what drives me. I love to make things. I don't particularly like showing them off. Whenever I'm at an event showing a work, I'm thinking I could be making something now! I think it might boil down to creating a complex but yet understandable response through computing structures to traditional artistic outputs.

While I'm thinking I'm very abstractly visual as in I draw structures. I'll be in the shower and I'll be drawing tiny abstract shapes of how I want a piece to flow and ebb in the steam on the shower screen. This is the same while I'm coding even for commercial things or while I'm doing art. I'm constantly sitting with a pad and I draw squares and maybe an arrow that points to something. I don't look back at that ever but for some reason it helps my brain download some of the thoughts into a representation that then locks that in so I can think about other things and it's still there. I think quite geometrically- shapes in a space with time animating through that space.

Q: Can you say something about how you write computer programs?

A: You think of each element of the program as being like a block, like a flow diagram. I think of programming in those terms, where you can see the whole picture of what you want to program, as a diagram and then you zoom in on that one little portion you are making and make that fit into the rest of the diagram. And in doing that it makes you reassess the diagram as a whole. I think that's one of my skills as a programmer: being able to put a complicated system into my brain and see how each component relates to it.

With the 3D vision work, for example, I want to put an image (a texture) onto a flat surface that can then fly around the room. I made a little texture engine that can load an image or a

movie. I build that module as simply and as flexibly as I can, then when I'm going to do the background elements, the pre-rendered elements, I'll re-use that element. I like to think in little modules and the benefit for me as an artist, is that I can then reuse things in a different context, in an unexpected manner. I do a lot of reusing and sharing of modules so I can try to promote obscure thinking… shoehorning a round design into a square problem space to see if that opens up new opportunities.

I have been programming since I was about four or five years old but the way I think about life and look at things in the world is almost using a programming logical flow. It's hard for me to separate my practice as an artist and the programming because I think programming has actually shaped my brain and the way it thinks about everything.

Q: Are there programming languages that suit different types of thinking?

A: I use visual languages like Max/MSP and Pure Data that work particularly well for sound because there is a quick turnaround and you can manipulate that in real time. Whereas even if you are a really quick coder you have to press compile and it takes a few moments to come up (with C++). If you are making audio Max/MSP has all the tools at hand whereas with programming you have to make a lot yourself. I've found when to comes to visual things I much prefer a scripted language or text based language where you can access each object or entity with much more clarity than a pipe line thing like Max/MSP. I feel that works for streams of data going to one or two or maybe ten sound streams or instruments if you like; whereas if you go visually there's a lot more data there and the stream based things get confusing more quickly…
Selecting the right tool (or tools) for the job is an important part of the creative process for me. I like to network a lot of my technology so that I can swap out languages or technologies for certain sections of the work if need be, if one language is better suited to a part of the work than another for whatever reason.

Q: Do you wait until you have done all the thinking about the nature of the program beforehand you start coding? Or do you code a bit and then go back to the overview?

A: I definitely code bit and then go back. There's several different ways. Sometimes you might go and see how the dancers are reacting to the system, but other times I jump in front and see what I think as a human and this back into the programming so I do iterate…

Q: What sort of things change as a result of looking at the dancers' movements?

A: It alters the way you see your end goal- what you think is interesting. I like to think about the motion tracking system we use with Stalker theatre which basically detects any kind of movement in the space. When you first go in there instantly you think, I want to track that dancer's movements and have the motion corresponding to that and I don't want to track the rope-like slings they use to perform. And then you start to look at the way it's working and the way they're using the slings and you realise, the slings are as much a part of this work as the humans are and we really should be tracking that movement as well. It makes you think, I'm going to this effort of separating this out for some logical reason that doesn't marry well with the art form. So, let's just rethink that, embrace that.

In *Creature: Interactions6,* I blended pre-rendered bush landscapes of trees, hills and rocks with interactive animal graphics to create a fantastical 'living' environment that the children interacted with. The problem with objects passing in front or behind trees is to do with compositing interactive graphics with pre-rendered graphics. In the 2D version, the interactive stuff was slapped on top of the pre-rendered graphics which gave it a designed look rather than a naturalistic look. In the 3D world, with the audience wearing 3D glasses I found that having incorrectly composited objects in 3D space gives you a headache. If a bird passes behind a tree, it needs to be obscured or hidden by that tree and if it passes in front of the tree,

it needs to obscure the tree as it passes in front. If this does not work correctly in 3D, your brain gets confused at the conflicting information.

Q: Did you discover that through the impact on you?

A: Oh yes. That's the way I work a lot. My art is predominantly interactive, so I create visuals and sounds that react to the performer or participant's movements. This adds a sense of discovery and play for the audience, I give them a virtual sandbox to play in and they can create their own fun and meaning out of that. But I always try things out for myself. I find research topics to look into by doing them first: for instance, I found out that the system I developed to optimise the 3D perspective on a cylindrical screen is called 'omnistereo rendering'[7] which already has research on it. I went into the Data Arena with its 360-degree cylindrical screen and projected using the normal rendering system and found the perspective was wrong. I came up with my own system and because I'd gone through all the effort to rectify it myself, I then had some context to help me find out if anyone else had the same problem. Creating a solution helps me understand the problem.

Q: If suddenly you couldn't program would there be an alternative?

A: That's a good question. When I started my sound and music design course after eighteen years of coding, I was thinking let's give music a go without computers, especially without coding. I still used digital audio tools which I consider to be a different process than using programming. The audio tools are more for composing music, putting pre-recorded sounds onto a time line, playing back, re-ordering them. Things that a traditional music composer would have done on pen and paper. I used to make one of these tools professionally as a software engineer and when I was making the tool I used to really love using it to make music even though I was a really bad musician. Basically, I got bored with having to sit at my desk coding making tools for other people to go and work creatively. My reaction to the sound and music course was to abandon coding to see what I could do as an artist- as a sound artist. After a year of this, I found the traditional tools to be quite limiting and started to combine my coding skills with my artistic vision and seem to have found my niche.

Q: What is the balance now between using digital tools and your creative coding?

A: I rely on a lot of digital tools. I do a little bit of 3D modelling, a little bit of Photoshop work, a little bit of illustration, music composition, music synthesizer. I consider them to be tools not the core. I consider the coding to be where my artistic strengths lie. If you were to tell me you've got one chance to make an innovative artwork and that's all you got, I would say my skill is in coding and I will use that. I think perhaps I've got a skill there that lets me explore things that haven't been explored much because there are not as many people with those skills artistically exploring random ideas. I feel there's a lot there to be explored that hasn't been done. There's such a history of pencil work, beautiful oil on canvas that is so hard to compete with. Whereas with this computational thing there is always some extra aspect to explore. You have the control to create whatever you want and that's what I love about it. So, if you are using one of these digital tools, you don't feel like you've got complete control to do what you want to do. They've got very hard restrictions that make it easier to do whatever the tool is designed to do whereas if you go into more low-level coding you can create whatever you want to create. You are restricted by hardware but it gives me freedom - I never picture exactly what I'm making while I'm making it so it gives me the freedom to not realize that while I'm creating it and then to see what comes out in the end.

Q: Are the tools partners in the creative process or is programming a part of you?

A: It's an interesting idea. Perhaps you could say that the underlying coding language is a part of you because you need to assimilate your own way of thinking with the flow of the programming language itself in order to effectively create large and complex applications. But it's not a one-way street, you also shape the program you are making to adhere to your own unique way of thinking. Then the software application that comes out of this coding, does act like a creative partner in an artwork. There is artistry and design on two separate

levels; there is artistry in creating an interesting entity and then there is artistry in partnering with it to create an actual artwork. When you are heavily involved in both stages, the trick is to spend at least as much time partnering as you do creating. This provides a richer knowledge of what might be interesting to add or change in the creation stage. I think an artist will always use or 'partner' with a software application in a slightly different way than the creator imagined, even when it's the same person doing both. I think it's important to be open and to embrace this.

Q: Does the programming lead to surprises or mistakes that are quite interesting?

A: There's been a few times when there's a mistake in the system. There's so much going on with the motion tracking across the stage and then going into fluid simulation that little logic mistakes can stay in there and you don't notice them for some time. A year later I've fixed a mistake that then has broken a nice effect we had. Then you think well how important is it to fix this mistake? More importantly is what part of that mistake caused something interesting and how can we extrapolate from the mistake and use whatever it was that caused the mistake and how can we design that in correctly into the system? An example is there was a gremlin in the particle system for almost three years, where occasionally old particles would become resurrected as black dots over the top of the new particles.Fixing the bug was surprisingly involved as it actually occurred on a few different levels, but once I had finally sorted it out, the show (*Bluespace*) started looking bland in a couple of scenes. I discovered that the bug was added a slight shading to the large particles that added a real sense of depth, almost a painterly feel to them and now that I fixed the bug they were looking very uniform, cold and digital. I ended up putting the bug back in as an optional feature.

Q: Do you deliberately look for surprises or unintended effects in the programming work?

A: I might design something to take parameters between zero and one and I deliberately throw in negative 5000 to see what happens. Or I deliberately connect the output of a visual camera into an audio synthesizer to see what happens. I throw nonsense at the system to see if interesting things happen. It's like designing for unintended usage not errors... You find something interesting as you're going along and you change your design with that and you'll find a little surprise and then you'll build that in. What you end up with out of the whole process is nothing like you wanted at the start…

Q. What kind of technical 'environments' do you find the most successful?

A: For me a successful environment will allow the programmer a lot of flexibility to perform whatever task they want, while providing access to an abundance of robust and useful pre-built scripts and minimising the effort needed to interface or connect these scripts together. These environments are often quite purpose built, and you will find environments that are specific for creating musical synthesizers, computer games, visual effects, controlling robots or any other number of creative tasks. These purpose-built environments will be quite efficient and easy to create objects within their own domain, but will often struggle to be used outside of this domain. The environment itself has its own affordances and desires which can often limit the scope of a developer, or nudge them in certain directions. This is what can give artworks made with a certain tool have a certain 'feel' or aesthetic. An example of this might be an abundance of artworks that used Photoshop filters to stylistically alter an existing photo. Generally, tools that are less specific provide more flexibility, but require more effort to actually do anything. In the world of software, C++ to me is such a tool, and it is why I like to use it, but I will commonly use collections of libraries to make creative visual coding easier, such as OpenFrameworks which contains many different camera and image processing routines.

A software environment will typically consist of an editor that is purpose built for working with the programming language itself (somewhat like 'Word' is designed for editing text and 'Photoshop' is designed for editing images). This editor will have in-built tools to make the process easier, such as a spell checker or thesaurus and will usually some form of in-built

documentation on the structure of the language itself and how to use it (somewhat like a dictionary). You will then find libraries of pre-written scripts which are easily downloadable. These scripts perform often repeated tasks such as ‘drawing an image to a screen’, ‘playing a sound file’ or ‘asking for some kind of textual input from the user’. The software environment will have some unified way to connect or ‘interface’ these existing libraries together and combine them with new purpose-built code. For me a successful environment will allow the programmer a lot of flexibility to perform whatever task they want, while providing access to an abundance of robust and useful pre-built scripts and minimising the effort needed to interface or connect these scripts together. These environments are often quite purpose built, and you will find environments that are specific for creating musical synthesizers, computer games, visual effects, controlling robots or any other number of creative tasks. These purpose-built environments will be quite efficient and easy to create objects within their own domain but will often struggle to be used outside of this domain. Generally, tools that are less specific provide more flexibility, but require more effort to actually do anything.

---

1 Photograph: Danielle Bluff

2 Andrew Bluff personal website: http://www.rollerchimp.com

3 Bluff, A.J. (2017). Interactive art, immersive technology and live performance. PhD Thesis, University of Technology Sydney: http://hdl.handle.net/10453/120340

4 Stalker: https://www.stalker.com.au/

5 Animal Logic Academy at University of Technology Sydney https://animallogicacademy.uts.edu.au

6 https://www.stalker.com.au/creature/

7 Simon, A. Smith, R.C. and Pawlicki, R.R. (2004). Omnistereo for panoramic virtual environment display systems. In Virtual Reality 2004. Proceedings IEEE, pp 67–279.