Creativity Versus the Perception of Creativity in Computational Systems

Simon Colton
Department of Computing
Imperial College, London
sgc@doc.ic.ac.uk

Abstract

We add to the discussion of how to assess the creativity of programs which generate artefacts such as poems, theorems, paintings, melodies, etc. To do so, we first review some existing frameworks for assessing artefact generation programs. Then, drawing on our experience of building both a mathematical discovery system and an automated painter, we argue that it is not appropriate to base the assessment of a system on its output alone, and that the way it produces artefacts also needs to be taken into account. We suggest a simple framework within which the behaviour of a program can be categorised and described which may add to the perception of creativity in the system.

Introduction

Increasingly, computer programs are written to perform tasks which, if undertaken by people, require an element of creativity. Within the computational creativity community, there are certain programs which are described as performing *artefact generation*, because they output valuable objects (artefacts) such as paintings, melodies, poems, theorems, etc., which can be assessed in their own right. While it is not a necessary requirement, there is an implicit assumption that to produce the most pleasing artefacts, aspects of human creative behaviour will have to be simulated. As a brief and incomplete survey, these programs work in domains of (i) literature and linguistics, e.g., poetry (Gervas 2000), story generation (Theune, Slabbers, & Hielkema 2007), joke generation (Binsted & Ritchie 1997), word invention (Veale 2006); (ii) music, e.g., composition (Baggi 1992), harmonisation (Phon-Amnuaisuk & Wiggins 1999); (iii) pure mathematics, e.g., theory formation (Colton 2002), conjecture making (Fajtlowicz 1988); and (iv) the visual arts, e.g., painterly renditions (Collomosse & Hall 2003), scene invention (McCorduck 1991), abstract art generation (Machado & Cardoso 2000).

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

We have developed two artefact generation programs. The first system, called HR, performs mathematical theory formation, and produces examples, concepts, conjectures and proofs. The second system, called The Painting Fool, is an automated artist which produces painterly renditions in a variety of ways. In both cases, assessment of the creativity of the system was a guiding factor in its development – we wanted to show an increase in creativity with each new version. In principal, there are two main factors we might take into account when we want to assess the creativity of an artefact generation program. Firstly, we can look at the output from the system, and assess the artefacts it produces. Secondly, we can look at the processes that the software performs, and assess its functionality.

As a research community, we have largely focused on assessment of creativity via assessment of the artefacts produced. We found that this was more than adequate in the development of HR, indeed a driving factor has always been a desire to increase the variety and quality of the mathematical artefacts it outputs. However, as we developed The Painting Fool, we began to understand that when consumers of paintings assess them, they do not strictly separate the process and the artefact. Indeed, their perception of how a piece of art was produced can have a major influence on their overall enjoyment of the artwork. This led us to consider more deeply how creativity in human artists is assessed. In particular, one might expect a consumer to assess the artwork and then project creativity onto the artist or not, depending on aspects of the artwork. While this model is certainly used in some situations, it other situations, more complex models are used. In particular, a consumer might endeavour to find out the process (intellectual, practical or otherwise) which was undertaken to produce an artwork. They might then use this information to make a judgement about the creativity of the artist, and finally use this judgement in their assessment of the artwork: a piece is better if it has been more creatively produced.

This more complex model of artistic assessment, and variants of it, seem prevalent in modern art circles. Indeed, in conceptual art, the aesthetic qualities of a piece seem to have little impact on a consumer's appreciation of the artwork. A classic example of this is Duchamp's displaying of a urinal as a piece of art. In situations like these, consumers are really celebrating the creativity of the artist rather then the value of the artefact. We argue that this complex model of artistic assessment is more pertinent when consumers assess the value of computer generated artworks. Hence, assessing software purely on its output is probably misguided. In response to this, we have developed a simple framework for managing the perception of creativity in the behaviour of software, and we describe this methodology here. To do so, we first give an overview of some methodologies for assessing the creativity of software, and then we describe some approaches to artefact generation in the visual arts. We then concentrate on how the creativity of visual arts software may be assessed, and we use this to present our approach to managing the perception of creativity. To highlight this approach, we present the HR system and The Painting Fool as case studies. While we have used the visual arts to highlight deficiencies in artefact-only based assessment methodologies, we conclude by arguing that our supplementary process-based approach may be used to drive the development of creative systems in general.

Assessment of Creativity in Software

Due to space considerations here, we discuss contributions by only three authors on the question of how to assess creativity in software. For a more comprehensive overview, we recommend (Ritchie 2007). Firstly, Boden not only suggested that, under certain circumstances, software could be considered creative, but also introduced a distinction in the assessment of artefacts, and a distinction in the assessment of the behaviour of creative systems (Boden 2003). In particular, Boden introduced the distinction between H-creative and P-creative artefacts: the former is novel to mankind, whereas the latter is novel only to the person/computer which discovered/invented it. Boden also introduced the distinction between exploratory and transformational searches for artefacts. In exploratory creativity, a search is undertaken within a well defined search space, whereas in transformational creativity, the space itself is modified. For a formal framework of creative processes which leads on from Boden's notions, see (Wiggins 2006).

Koza is well known for using evolutionary approaches to produce artefacts such as circuit board designs which are patentable. As such, while he doesn't make claims of creativity directly, his software could certainly be considered so. In (Koza *et al.* 2003), the authors justify their statement that *"genetic programming now routinely delivers high-return human-competitive machine intelligence"*, by explaining the terminology expressed. In particular, they touch on aspects of artefact-based assessment, such as whether the result is competitive with those produced by a human, as evidenced by the re-invention of patentable ideas, or the invention of new patentable ideas. They also touch on process-based assessment, by describing the usage of software as 'routine' if it requires little or no tweaking to work on a new application. For more discussion of fine-tuning, see (Colton, Pease, & Ritchie 2001).

The most extensive contribution to the discussion of how we could assess creativity in software has come from Ritchie. Over a series of papers, culminating in (Ritchie 2007), he has introduced a framework for assessing the creativity of an artefact generation program. He starts with some clear assumptions, including relying only on the artefacts produced by the system to assess its creativity. He states that:

> "*For the purposes of setting up an initial framework, we shall adopt the (possibly over-simplified) assumption that the internal workings of a program are not part of the relevant data.*"

Ritchie claims that the creativity of an individual is manifest in the artefacts they produce, so we can ignore the process behind the production of the artefact. He adds that *"this may be our most contentious working assumption"*, and gives these reasons for the simplification: (a) the creativity of humans is normally judged by what they produce, and so this gives a level-playing field when assessing artefacts produced by humans or computer (b) underlying processes are not observable factors, hence not reliable, and (c) there is a risk of circularity in the argument if we assess both artefact and process. This last point is quite subtle: Ritchie suggests that we should think of innovation in a production method as the generation of an abstract artefact. As this manifests creativity which can be assessed, we only need to consider the assessment of creativity via artefacts, not processes (and if we consider processes separately, we may make the argument circular).

Ritchie develops his framework to capture two intuitively key properties of artefacts, namely their quality and their novelty. With respect to quality, he explains that in certain domains, most notably computational literature, many systems aren't able to reliably produce artefacts which satisfy the *definition* of the class of artefacts they should belong to, e.g., joke generators often produce sentences which would not be recognised as jokes. Hence notions of class membership and the quality of class members are taken into account in his framework. In terms of novelty, Ritchie uses the notion of an inspiring set of artefacts. This set is used in the development of the software, hence the output of these artefacts should not be seen as a great success. For instance, the development of each concept production rule in our HR system (Colton 2002) was inspired by one or two concepts which we wanted HR to re-invent. The fact that each production rule was subsequently used to discover dozens of interesting concepts outside the inspiring set adds to any claims of creativity for HR, and Ritchie captures this idea in his framework. As it is well developed, Ritchie's framework has been applied to certain creative systems, e.g., (Gervas 2002). Details of these applications are given in (Ritchie 2007), as are some suggestions for extending the framework.

Artefact Generation in the Visual Arts

There are a myriad of ways in which visual artefacts have been produced, and, as computer graphics is a huge area, for space considerations, we discuss only the most pertinent subareas: evolutionary art, Non-photorealistic Rendering (NPR), and automated painting. The aim of evolutionary art projects is largely to generate abstract images, by enabling the evolution of programs which generate the images. The evolution is usually guided by the user making aesthetic preferences amongst the phenotypes (images) generated by the genotypes (programs) he/she is presented with. Their choices inform the production of new genotypes via the crossover of material from parent programs into offspring, and the phenotypes evolve accordingly, until an artwork which the user is happy with emerges. An important project within evolutionary art has been Latham et. al's development of the Mutator system (Todd & Latham 1992). Also, Machado et. al's NEvAr evolutionary art system is distinguished by it being able to work fully automatically using a fitness function (Machado & Cardoso 2000). Our contribution to this area is described in (Hull & Colton 2007).

The aim in Non-Photorealistic Rendering (NPR) is, broadly speaking, to produce images that look like they may have been painted/drawn/sketched by a human artist. For instance, numerous implementations can turn a digital photograph into a passable simulated impressionistic painting, e.g., (Litwinowicz 1997). Some of the techniques used in NPR methods include segmentation (turning an image into a relatively small number of regions of colours) and the simulation both of natural media such as paints, charcoals and pastels, and their usage, e.g., painting with a brush, smudging charcoals, etc. NPR methods have become highly sophisticated. A good example is the work of Collomosse et. al, for instance: the usage of saliency maps to enable fine detail painting of regions of interest in an image, e.g., facial features (Collomosse & Hall 2006); and the simulation of cubist renderings (Collomosse & Hall 2003).

NPR systems aim to simulate the *results* of human painting without necessarily having to simulate the painting process itself. While some physical aspects are simulated, e.g., the placing of paint strokes, most NPR software doesn't simulate the many cognitive aspects of the artistic process, such as choosing subject matter and painting style to embed a concept, scene invention, abstraction, attention to detail, etc. Human artists regularly use simulated paint brushes, such as those within Adobe Illustrator to produce their artworks, e.g., (Faure-Walker 2006). When doing so, they worry more about higher level details (subject matter, abstraction, scene layout, etc.) than what an individual paint stroke should look like, or the physics underlying paint flow. When software is written to use similar tools and to consider similar high-level details to automatically generate entire paintings, we call these programs automated painters. The most well known program in this area is AARON (McCorduck 1991), which has been developed by artist Harold Cohen over dozens of years to simulate aspects of his own painting process. These aspects include colour and abstraction choices, the invention of scenes including people and objects in rooms, and the painting of large colour regions. Our contribution in this area has been The Painting Fool project (described below). While NPR projects aim to produce pleasing images which could have been painted by humans, we want The Painting Fool to produce pleasing images which *couldn't* have been painted by humans.

A Note on Art Appreciation

In a simple appreciation model, a consumer likes the paintings of a particular artist, then – maybe over a period of time – the consumer bestows the label of creativity onto the artist as they begin to appreciate the artist's aesthetics, style and innovative methods. It is important to note that there is no collective notion of beauty within art intelligencia. Moreover, if there were such a collective notion, artists would willfully rebel, and would be expected to do so. Such willful disrespect of collective notions of beauty within the wider community are commonplace by art critics and artists alike, as witnessed by the critical dislike of the hugely popular artist Jack Vettriano, and by the production of shock art pieces, exemplified by the sculptures of the Chapman brothers. Artists generally want to show a progression in their work, and so they change their own aesthetic considerations, along with their painting style, subject matter, etc., as their career progresses. For instance, talking about Willem de Kooning (whose style eventually became extremely non-representational), Rudolph Burckhardt recalled that:

> "*Once he said he'd like to paint like Ingres and Soutine... He made a few exquisite, Ingres-like drawings... but then he said that if he kept this up he'd go crazy.*" *[(Hess 2007), page 11].*

We contend that while this simplified view of art/artist appreciation may be the default in society, there are other, more complex models of art appreciation. In particular, a model of real relevance to computer generated art is as follows: the consumer endeavours to discover the process behind the production of a particular artwork by a particular artist. They then make various judgements about, amongst other things: (a) the effort behind the process (b) the ingenuity in devising the process, and (c) the skill required to undertake the process. These judgements – possibly accompanied by an aesthetic judgement – are used to determine how much the consumer likes the piece.

Imagine an art-lover at an exhibition entitled *'Dots 2008'*. He speaks to two artists, each displaying a painting. In both cases, the art-lover cannot see past the seemingly random arrangement of dots of paint. He mentions this to the first artist, who says: *"Oh, no, they're not randomly placed. Each dot represents a friend of mine. The colour of the dot represents how I feel about them, and the position indicates how close I am to them."* The art-lover moves on to the next artist, and mentions again that the dots look like they have been randomly placed. The artist replies: *"Yes, that's right – I just mixed a random colour and dabbed it onto the canvas"*. Returning a week later with a friend, our art-lover wants to purchase the first painting, explaining to the friend that it represents feelings. Neither artist is present, and the art-lover cannot remember which painting is which. The friend points out that perhaps they could work out which one represents feelings, but they fail to see anything but randomness in both works. Finally, when the friend points out that both paintings look alike, so they should just choose one, our art-lover is inconsolable, and buys neither.

We believe that this situation is plausible, and more importantly, that it is perfectly reasonable for the art-lover of this story to prefer one painting over another, even though they are so similar on the surface. We are in an age where the photograph has largely replaced fine art for representational purposes; an age which has seen various movements in art such as impressionism, cubism, abstract expressionism and many more, which were instigated by artists moving from being craftsmen to intellectuals who use artistic techniques as their medium of expression. It is therefore not difficult to see why, in many circumstances, the creativity of an artist is a primary consideration, with the beauty of their work being secondary. At one end of the scale, conceptual art embodies this the most explicitly:

> *"Conceptual art is not about forms or materials, but about ideas and meanings ... In particular, conceptual art challenges the traditional status of the art object as unique, collectable or saleable." [(Godfrey 1998), page 4].*

Even at modest positions on the scale, artists are expected to create both at the conceptual and the craft level, and art-lovers are expected to appreciate both. In many cases, the conceptual innovation will occur at a level where it is exhibited visually in the artwork. In other cases, as with the *'Dots 2008'* artist, innovation may occur in a way that it is not obviously exhibited in the artwork, but was exhibited in the process.

The Perception of Creativity in Software

We can use Ritchie's idea of considering an artistic process as an abstract artefact which manifests the creativity of its inventor, to explain that, in the story above, the art-lover's overall aesthetic includes aspects of the creativity behind the process, and they have identified more creativity in the first artist than in the second. We believe that it is not just creativity that the art-lover is looking for in the process, but also effort and skill, and possibly many other aspects. Even if this is not the case in general, we can still conclude that the assessment of an artwork can include information about the artistic process behind it. This adds to a number of difficulties associated with computer generated art. The first of these difficulties is that artists using computers in any fashion tend to be kept as outsiders in the art world. This is probably due to a general reluctance to admit that hard-learned traditional crafts can be replaced with digital substitutes, and possibly the incorrect perception that engineering skills such as computer programming do not lend themselves well to artistic expression. This has left computer-based artists feeling like outsiders, although this suits some people (Paul Brown, personal communication).

Computational creativity in the visual arts causes even more concern in the art world. Fortunately for them, however, the default position has always been resolutely negative on this point. For instance, in the June 1934 edition of the Meccano magazine, an article entitled 'Are Thinking Machines Possible?', reporting on some meccano machines able to solve mathematical equations, concluded by stating that:

> "*Truly creative thinking of course will always remain beyond the power of any machine.*"

We can add to this a general reluctance in the non-photorealistic rendering (NPR) community to discuss issues of creativity. Indeed, some researchers seem almost apologetic for simulating processes too close to human creativity. For instance, in a key textbook on NPR techniques, the authors state unequivocally that:

> "*Simulating artistic techniques means also simulating human thinking and reasoning, especially*

creative thinking. This is impossible to do using algorithms or information processing systems." [(Strothotte & Schlechtweg 2002), page 113].

This leaves the general impression with art consumers that computers aren't and will never be creative. This wouldn't be a problem if the artworks produced by computers were assessed in their own right, but we have argued that the process – and in particular the creativity – underlying the process is taken into account in this assessment. This leads computer generated art into a vicious circle: the default position that software is not creative leads to a low assessment of an artefact which it produces, but then if the software produces bad artefacts, it really cannot be creative. As an aside, the idea of a Turing-style test for computer generated art is often touted: can people tell which of two paintings was computer generated, and which was painted by a human artist? This is asking the wrong question. A more pertinent question would be: which artwork would people buy? In this case, to simulate a real gallery situation, full disclosure of the origin of each work would be required. In the current climate, for good reasons, this would not favour the computer artist, as its (seemingly) uncreative artistic methods would go against it.

Our first reason to consider how people perceive the behaviour of software is therefore to try to break the vicious circle and enable computer generated art to compete on a level playing field with human art. Our second reason is that ultimately, automated painters should – like human painters – transcend any information they are given about the nature of good and bad artefacts, and should develop their own aesthetic along with their own styles. Hence, fixing measures of how good or bad generated artefacts are is missing the point in the visual arts. This is only useful in the long run for software where there is no ambition for it to be creative in its own right (as is the case in most NPR applications).

The Creative Tripod

To begin to prescribe how we might portray the behaviour of artefact generation processes, we first note that under normal circumstances, only the artist and, in some situations the audience, can have creativity attributed to them. When computers are used, however, it is commonplace for people to attribute creativity to the programmer in addition (or instead of) the software. This could be seen as a double standard, as creativity wouldn't ordinarily be attributed to the teacher of a student who produced an artwork, but because the training of software is far more explicit than that of an art student, it is understandable. It is our responsibility to point out that the inclusion of random processes and the alteration of code through evolutionary means and/or machine learning methods will often mean that the behaviour of the software cannot be predicted by the programmer (which is the effect we are looking for). We must also note that there is a potential dichotomy in explaining computational processes: too little information will not feed the desire to understand what it is doing, but too much information might re-inforce the impression that the software is purely carrying out predefined (programmed) instructions.

Given the default position in the popular perception of machines that software cannot be creative, we can expect a certain amount of criticism towards any implication that software is being creative. We can identify such criticisms as a set of necessary conditions, and manage them in our portrayal of the software processes. We propose to concentrate on three such necessary conditions, namely that the software exhibits behaviour which could be described as *skillful, appreciative* and *imaginative.* One can imagine a painter (human or otherwise) who lacks one of these three behaviours. Without skill, they would never produce anything; without appreciation, they would never produce anything of value; and without imagination, at best they would only produce pastiches of other people's work.

To aid in describing the behaviour of creative software in straightforward terminology, we introduce the notion of the **creative tripod**. The three legs of the tripod represent the three behaviours we require in our system: skill, appreciation and imagination, and only if all of these are present will the tripod support the perception of creativity. Moreover, the legs of tripods are extensible and are themselves split into three sections. We use this to highlight that there are three parties which could be perceived as contributing creatively when a consumer experiences a computer generated artwork, namely the programmer, the computer and the consumer. Moreover, each party can contribute skill, appreciation and imagination to the experience, and the relative extension of the nine sections spread over the three legs can be used to represent the relative size of the contribution. Our position is that, if we perceive that the software has been skillful, appreciative and imaginative, then, regardless of the behaviour of the consumer or programmer, the software should be considered creative. Without all three behaviours, it should not be considered creative, but the more aspects which extend each leg of the tripod, the more creativity we should project onto the software.

This analogy provides both a very straightforward way of categorising and describing the behaviours of creative software for non-technical consumers, and a way of assessing the amount (if any) of creativity exhibited. We can use the framework to make concrete decisions about the creativity of pieces of software. Looking at the AARON program, we note that it doesn't have any notion of the value of its own artwork. Cohen might investigate using machine learning techniques to derive some rules governing which of AARON's paintings he keeps each morning when he checks the output, but this seems unlikely to happen (Harold Cohen, personal communication). Hence, in our perception of how AARON works, it is difficult to describe any of its behaviours as appreciative. So, even though we could describe its scene generation and painting of the scenes as imaginative and skillful respectively, we cannot call the software creative. Conversely, the saliency detection behaviour of Collomosse's cubist image generation software (Collomosse & Hall 2003) can be described as an appreciation of the important aspects its subject matter, and the ability to construct cubist representations from a digital photograph could be described as (perhaps mildly) imaginative. Given the skill that the software has in simulating paint strokes to render the image, we would be prepared to describe this software as creative. For similar reasons, working in fully automated mode, we would use the word creative to describe the NEvAr system – in this case, its fitness function provides appreciation that evolutionary art programs rarely have.

We can also refer back to other frameworks for assessing creativity in software. In particular, one might consider that software performing exploratory search in Boden's terminology should be described as less imaginative than software performing transformational search. We might also look at Koza's notion of software performing 'routinely', and say that the lack of fine-tuning required is an indicator of greater appreciation of the software to adapt to new applications. Finally, one might consider that when assessing the creativity of a program, a simple framework in which the processes behind artefact generation are considered in informal terms is a good complement to Ritchie's framework which considers the artefacts generated in more formal terms. Moreover, we argue that we should be implementing Ritchie's criteria into artefact generation software, so that it could better *appreciate* its own creativity.

Case Studies

The HR system is named after mathematicians Hardy and Ramanujan, and has been designed to form theories in domains of pure mathematics. HR's input is typically minimal information about a domain, such as the axioms of an algebra like group theory, or some simple concepts such as the arithmetic operators in number theory. HR operates by building new concepts from old ones using a set of production rules which essentially impose structure or constraints on the examples which satisfy the definition of the new concept. In addition to inventing new concepts, HR also looks for empirical relationships between the concepts. For instance, if the examples of one concept are exactly the same as those for another, HR makes a conjecture that the two concepts are logically equivalent. HR then employs third party automated reasoning software such as Otter (McCune 1990) and MACE (McCune 1994) to try and prove (respectively disprove) that each conjecture follows from a set of user supplied axioms. The heuristic search for concepts and conjectures is driven by a set of *measures of interestingness*, as described in (Colton, Bundy, & Walsh 2000). That is, at each round, the most interesting concept – as specified by a user given weighted sum of measures – is identified and then new concepts are built from it. HR has been quite successful in various domains of pure mathematics, including number theory (Colton 1999) and algebraic domains (Colton *et al.* 2004). A formal description of HR, and a summary of some of the mathematical applications to which we have applied HR can be found in (Colton & Muggleton 2006), and a more extensive description can be found in (Colton 2002).

HR was developed before the creative tripod framework. In fact, we developed HR strictly according to an assessment scheme involving only the artefacts. It didn't matter whether a description of our techniques would add to the perception of creativity as a whole or not, as long as each new version of HR produced better quality and/or more types of mathematical artefacts. We have been prepared to use the word 'creativity' to describe HR for a number of years, and we can now justify this (somewhat) by appealing to the creative tripod. HR's skills include its ability to form new concepts from old, to make conjectures empirically and to prove/disprove theorems (including Otter/MACE as a process in HR's theory formation). HR's measures of interestingness enable it to appreciate the concepts and conjectures in the theory it is producing, and to change its behaviour accordingly. We can describe the search that HR performs as imaginative, for example, we can give HR just the ability to multiply two numbers together, and come back ten minutes later to find that it has discovered that odd refactorable numbers are perfect squares (refactorable numbers x are such that the number of divisors of x is itself a divisor). If we gave a child (or even an undergraduate) the same ability to multiply two numbers and 10 minutes, and it produced the same result, we might be inclined to describe him/her as imaginative (or at least inventive). While refactorable numbers were a P-creative invention (Colton 1999), the conjecture above was (we believe) a H-creative invention, which adds weight to our claim of imaginative behaviour on HR's part.

The second system we have built for artefact generation is called The Painting Fool. This produces pieces of visual art, such as (simulated) paintings, sketches and drawings. To do this, it simulates both natural media such as pens, pencils, brushes, canvases, paints, etc., and it simulates the way in which those media could be used. In particular, working from a given digital image, it first segments the image into paint regions, and abstracts the borders of the regions to be fairly smooth. Then, working in painting layers, it simulates different methods for using natural media to both outline and fill each region. In addition, The Painting Fool has been trained with a knowledge base of different settings for the segmenting and rendering processes, and it has an expert system which is able to take high level information in the form of keywords and map these to settings. In effect, this enables keyword control of all aspects of the artistic style it uses for a particular artwork, including the level of abstraction, the colour palette, the natural media which are simulated and the painting style. Details about The Painting Fool have not yet been published, but there is a web page devoted to it here: `www.thepaintingfool.com`.

The Painting Fool was developed very much as an art project rather than a scientific project. That is, no technical papers about it have been published (although one has been recently submitted to a conference). Instead, we have opted to attempt to disseminate artworks by The Painting Fool, by preparing online galleries and publicising them, taking part in exhibitions, giving demonstrations, entering competitions, etc. In addition, The Painting Fool has been designed to operate in real time, i.e., it adds each stroke to a canvas with painstaking detail. We have added videos of this process to the web site, so that people can see it working. All of these decisions were taken to aid in presenting The Painting Fool as an artist rather than a program, and the website is written in The Painting Fool's voice to further emphasise this. The dissemination of its artwork was in order to receive critical feedback about it as an artist and about the paintings it produces. The feedback has been generally positive. Although we have only anecdotal evidence, it is apparent that being able to watch The Painting Fool create its paintings means that people project more value onto them than they would if the paintings were rapidly generated through, say, an image filtering process. This seems to be because they can project critical thought processes onto the software, and empathise with it more. We also believe that the description of its behaviour in terms of skill, appreciation and imagination helps to add to the empathy people have for The Painting Fool.

The development of The Painting Fool has been heavily influenced by the creative tripod framework. The first gallery we displayed was designed to stretch its skills: a series of nine cityscape and building paintings/sketches, which required the implementation of abilities to simulate various styles in acrylic paints, pencils, pastels, charcoals and chalks. The second gallery was designed to increase The Painting Fool's appreciation, both of its subject matter and the way in which its painting materials and styles can affect the artefacts it produces. The gallery contains 222 portraits of Audrey Tatou from the film *Amelie*, and in each one she is expressing an emotion. We added around fifty artistic styles to The Painting Fool's knowledge base and tagged each one with an emotion keyword. The keywords could then be used as a high level way to choose settings for it to paint with. For instance, there is a mapping of the keyword 'happy' onto a vivid colour palette and a slapdash painting style which simulates acrylic paints.

At this stage, the Painting Fool was able to act like an expert system, but needed to be told the emotion expressed in the image to be able to choose from its knowledge base accordingly. To further increase its automation, and to increase the perception of the appreciation it has, we combined it with a machine vision program which is able to detect the emotion that a person expresses in a short video clip (Valstar & Pantic 2006). The vision system passed the emotion keyword to The Painting Fool, which chose its artistic style accordingly when painting a portrait of the person in the video clip. We entered this combined system into (and won) the British Computer Society's annual Machine Intelligence competition.

Driven by the desire for The Painting Fool to exhibit skill, appreciation and imagination, we have started to give it abilities which might be described as imaginative. In particular, we wrote a scene generation module that uses an evolutionary approach to build scenes containing objects of a similar nature, such as city skylines and flower arrangements (Colton 2008). Taking the approach to a meta-level, we combined the system with HR, which we used to generate fitness functions, so that it produced scenes which maximised a fitness function that we had not explicitly specified. This added greatly to our perception of imagination in the system. At present, The Painting Fool does not exhibit behaviour which is skillful, appreciative and imaginative at the same time, but we are working on a project where a natural language module will be used to parse some text, and then construct a scene which The Painting Fool will paint. We hope this will exhibit multiple behaviours on each leg of the tripod, and that we will be able to describe The Painting Fool as creative.

Conclusions

Consideration of how to assess the level of creativity of an artefact generation program is a key issue in computational creativity. Most work in this area has concentrated on how to assess creativity using only the artefacts produced by the system. We have argued that – at least in the visual arts – the process of creating an artefact is often a deciding factor in the assessment of that artefact. This causes a problem in computer generated art, because the general impression is that computers cannot be creative, and hence assessment of their artwork suffers. We have also described other problems related to assessing artefacts in a domain where the assessment criteria are themselves in flux. Moreover, we have pointed out that software which we want to ultimately be accepted as creative in its own right needs to subvert any given notions of good and bad artefacts.

Given these considerations, we have argued that we should be providing consumers of the artefacts with some high-level details of how the software operates. By introducing the creative tripod for this purpose, we hope to be able to level the playing field somewhat when people assess the value of computer generated artefacts. Moreover, we hope that developers of creative systems in future will refer to the creative tripod to estimate whether consumers will perceive their software as creative or not. We further hope that developers will improve their system to exhibit behaviours which are skillful, appreciative and imaginative. We hope to have shown that in certain domains, not least the visual arts, considerations about how creative software is perceived to be are as important as aesthetic considerations, and how creative the software actually is.

References

Baggi, D., ed. 1992. *Readings in Computer Generated Music*. IEEE Computer Society Press.

Binsted, K., and Ritchie, G. 1997. Computational rules for generating punning riddles. *International Journal of Humour Research* 10(1):25–76.

Boden, M. 2003. *The Creative Mind: Myths and Mechanisms (second edition)*. Routledge.

Collomosse, J., and Hall, P. 2003. Cubist style rendering of photographs. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 9(4):443–453.

Collomosse, J., and Hall, P. 2006. Salience-adaptive painterly rendering using genetic search. *Intl. Journal on Artificial Intelligence Tools (IJAIT)* 15(4):551–576.

Colton, S., and Muggleton, S. 2006. Mathematical applications of Inductive Logic Programming. *Machine Learning* 64:25–64.

Colton, S.; Meier, A.; Sorge, V.; and McCasland, R. 2004. Automatic generation of classification theorems for finite algebras. In *Proceedings of the International Joint Conference on Automated Reasoning*, 400–414.

Colton, S.; Bundy, A.; and Walsh, T. 2000. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies* 53(3):351–375.

Colton, S.; Pease, A.; and Ritchie, G. 2001. The effect of input knowledge on creativity. In *Proceedings of the ICCBR'01 Workshop on Creative Systems*.

Colton, S. 1999. Refactorable numbers - a machine invention. *Journal of Integer Sequences* 2.

Colton, S. 2002. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag.

Colton, S. 2008. Automatic invention of fitness functions, with application to scene generation. In *Proceedings of the EvoMusArt Workshop*.

Fajtlowicz, S. 1988. On conjectures of Graffiti. *Discrete Mathematics 72* 23:113–118.

Faure-Walker, J. 2006. *Painting the Digital River*. Prentice Hall.

Gervas, P. 2000. Wasp: Evaluation of different strategies for the automatic generation of spanish verse. In *Proceedings of the AISB-00 Symposium on Creative and Cultural Aspects of AI*.

Gervas, P. 2002. Exploring quantitative evaluations of the creativity of automatic poets. In *Second workshop on creative systems, approaches to creativity in Artificial Intelligence and Cognitive Science (ECAI)*.

Godfrey, T. 1998. *Conceptual Art*. Phaidon.

Hess, B. 2007. *De Kooning*. Taschen.

Hull, M., and Colton, S. 2007. Towards a general framework for program generation in creative domains. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*.

Koza, J.; Streeter, M.; Mydlowec, W.; Yu, J.; and Lanza, G. 2003. *Genetic Programming IV*.

Litwinowicz, P. 1997. Processing images and video for an impressionist effect. In *SIGGRAPH '97: Proceedings of the 24th annual conference on computer graphics and interactive techniques*.

Machado, P., and Cardoso, A. 2000. NEvAr – the assessment of an evolutionary art tool. In *Proceedings of the AISB00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

McCorduck, P. 1991. *AARON's Code: Meta-Art, Artificial Intelligence, and the Work of Harold Cohen*. W.H. Freeman and Company.

McCune, W. 1990. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories.

McCune, W. 1994. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories.

Phon-Amnuaisuk, S., and Wiggins, G. 1999. The four part harmonisation problem: A comparison between genetic algorithms and a rule-based system. In *Proceedings of the AISB'99 Symposium on Musical Creativity*.

Ritchie, G. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines* 17:67–99.

Strothotte, T., and Schlechtweg, S. 2002. *Non-Photorealistic Computer Graphics*. Morgan Kaufmann.

Theune, M.; Slabbers, N.; and Hielkema, F. 2007. The automatic generation of narratives. In *Proceedings of the 17th Conference on Computational Linguistics in the Netherlands (CLIN-17)*.

Todd, S., and Latham, W. 1992. *Evolutionary Art and Computers*. Academic Press.

Valstar, M., and Pantic, M. 2006. Biologically vs. logic inspired encoding of facial actions and emotions in video. *IEEE International Conference on Multimedia and Expo* 325–328.

Veale, T. 2006. Tracking the lexical zeitgeist with wordnet and wikipedia. In *Proceedings of the 17th European Conference on Artificial Intelligence*.

Wiggins, G. 2006. Searching for computational creativity. *New Generation Computing* 24(3):209–222.