Casual Creation, Curation, Captioning, Clustering and Crossover in the Art Done Quick App

Simon Colton
SensiLab, Monash University, Melbourne, Australia
Game AI Group, EECS, Queen Mary University of London, UK

Abstract

We report on the latest developments with the Art Done Quick casual creator app, which enables users to rapidly explore a space of generated images, edit, personalise and share particular images, and curate them on a large sheet. This has been developed with a fun-first methodology which has thrown up a number of interesting new research problems, but also led to the solution of a long-standing issue with the crossover of images in Art Done Quick, which at its core is an evolutionary art system. We describe advances in enabling users to create more varied and interesting images, methods to enable them to curate and cluster images, a pseudo-bullshit caption generator for fun feedback, and how we have changed the genome representation to produce messier images via much more satisfying crossover operations.

Introduction

Art Done Quick is a casual creator app in the sense described in (Compton and Mateas 2015) and expanded in (Compton 2019), as a creativity support tool for amateurs, where user enjoyment is prioritised over precision control, sophistication of output and professional usage. It is an iOS app for touchscreen devices that enables users to explore a space of decorative images in two modalities. The first of these is generative image evolution, with a *sheet interface* where users randomly generate, mutate and crossover images. The second enables graphic design via an *edit interface* that allows users to post-process an image with various image filters, collaging techniques and drawing tools. As described in detail in (Colton et al. 2020a), Art Done Quick has been developed from the basis of an evolutionary art tool (Romero and Machado 2007) with a *fun-first* methodology. This has largely entailed (a) implementing casual creator design patterns from (Compton and Mateas 2015), e.g., by giving users fun feedback about their creations (b) identifying and minimising frustrations in the evolutionary art process, e.g., by decreasing image generation times, and (c) opportunistically implementing fun techniques that users enjoy in other casual creators, e.g., enabling the drawing of mandala-like designs.

At every stage in the development of the app, any new functionality and/or user experience has been challenged with the question: "will this make the app more fun to use?" rather than "will this produce more sophisticated imagery?" or "will this enable finer-grained control?" For instance, techniques for altering images were largely chosen from those which have historically been more for fun than for practical application, such as liquifying, photo montaging and pixelating. Also, as people enjoy taking ownership of their work, tools for adding text, stickers and drawings were implemented, so users can personalise their creations.

In general, users of the app employ the sheet interface to rapidly produce *raw-material* images that stand alone as interesting and aesthetically pleasing artefacts, but can also become the basis for post-processing design via the edit interface. The sheet has $70 \times 70 = 4900$ empty cells available for users to tap on, which populates the cell with a randomly generated 500×500 pixel image. Double tapping on an image will produce 8 variations of it in neighbouring cells, with existing images moved out of the way to make room. Dragging and dropping one image onto another likewise produces 10 offspring children in nearby cells. Users can tap an image to see it re-rendered at 2000×2000 pixels, and from there they can invoke the edit interface to alter the image.

Images are generated via a particle-based approach which involves mathematical functions guiding the movement and colour of up to 600 particles over a series of up to 100 time steps. Shapes are rendered at the particle positions, in the particle's colour at each time step, and blurring happens every ten time steps. The functions which initialise the particles form the `init` gene in the chromosomal (in evolutionary art terms) representation of the image. The gene contains five *base* functions, which determine the (x, y) locations and (red, green, blue) colours for each particle, based on the particle's ID number. The functions which guide how the particles change over time form the `update` gene, and comprise ones for calculating (x, y, r, g, b) as before. A third gene in the chromosome, `render`, consists of bases such as transparency and size, which determine how the shapes are rendered at the particle locations, and how a *symmetry transform*, such as a kaleidoscope action, further re-arranges the particles. The final gene in the chromosome, `postprocess`, holds information about a series of transformations that the user has imposed on the raw-material image via the edit interface. Prior to the work presented here (see the section on crossover below), image generation was controlled by a single chromosome per image, but now the genome representation can contain multiple chromosomes.

We describe here the latest developments in the Art Done Quick project. In the next section, we look at the ways in which users can create new imagery through enhanced editing tools, how they can collate and curate their creations, and how we have added a pseudo-bullshit text generator to provide fun captions for images. Following this, we look at how the app can automatically cluster large sets of images, to help users arrange their art on the sheet interface. We then describe a new crossover operation that solves the long-standing problem of unsatisfying results arising from crossing over two images. We end by suggesting some general guidelines for casual creation design arising from this work, and describing some future directions for the project.

Creation, Curation and Captioning

A major milestone was reached in early 2020 when the edit interface to Art Done Quick was finished. This greatly expanded the range and sophistication of the images that users can make with the app, and – importantly – also increased the amount of fun users can potentially have while making art, as image filtering, collaging and drawing are engaging and amusing pastimes over and above their use in producing images. The images in figure 1 were produced using the edit interface, which comprises the following ten toolsets, all of which alter the chosen image in real-time.

- **Art**: where the shape type, size and transparency bases, the symmetry transform granularity and the number of time steps and particles from the `render` gene are altered, with the raw-material image re-rendered accordingly.
- **Effects**: where users apply edge detection, blurring, pixellation, dithering, styles and montaging of various types.
- **Colour**: where users tint and posterize images, add rainbow effects and change saturation levels.
- **Texture**: where users add artistic, grungy, material and patterned texture overlays.
- **Liquify**: where users liquify images with holes, glass balls, twirling, squeezing, bulging and carnival mirrors.
- **Light**: where users change ambient lighting, add spotlights, floodlights and shadows, and make the image glow.
- **Montage**: where users produce kaleidoscope, spiral, compound eye, ring, decreasing-circle and explosion versions of their image, optionally overlaid in multiple layers.
- **Stickers**: where users add stickers to their image, in various styles and colours, with optional symmetrical repetition.
- **Draw**: where users can use simulated paint brushes, pens, pencils, crayons, highlighters and light pens to draw over or under their image, producing mandalas if they want to.
- **Words**: where users can add plain, bordered, patterned, bubbled, shadowed and glowing texts, in various fonts and paragraph styles, to their image.

The speed at which images can be generated has improved greatly in recent developments, with Art Done Quick taking advantage of multiple cores on iPad and iPhone devices, as described in (Colton et al. 2020a). This means that users can fill the sheet to capacity (1,000 images on an iPad Pro, 500 images on other devices) in minutes, which has necessitated tools to enable users to organise and curate the artefacts they produce. Most important amongst these tools is automated clustering, as described in the next section. In addition, the app enables users to drag images around on the sheet to curate their own collections, to batch-select images for deletion or to employ a *Thanos* effect where a random 50% of the images are deleted. Users can also *like* images, save images to their camera roll, and invoke slideshows of liked, randomly chosen or clustered images.

As depicted in the first (sheet interface) and second (edit interface) screenshots of figure 2, and described in (Colton et al. 2020b), we have implemented a secondary AI system which controls a virtual hand that acts like (human) users of Art Done Quick. This explores the space, mutates chosen images and edits them to produce ones predicted with high confidence to contain particular objects/scenery by the ResNet50 machine vision model (Krizhevsky, Sutskever, and Hinton 2012). Originally intended to illustrate opportunities in going from casual to computational creativity, we are currently adapting it to act as an auto-pilot to entertain users while showing them the app's capabilities.

Another interesting addition to the app has been image captioning, whereby Art Done Quick produces possibly amusing and/or inspiring feedback for users in terms of a caption for images, as portrayed in the third screenshot of figure 2. As described in (Colton, Berns and Pérez-Ferrer 2020), this also uses ResNet to predict the content of images, as such false positives act like an interpretation of the abstract imagery. This information, supplemented by a colour profile of the image, is employed via text templates using International Art English (Rule and Levine 2012) to generate a caption for a user's image, if they zoom into it on the sheet interface. When ResNet doesn't predict any content, the system generates pseudo-bullshit (Pennycook et al. 2015) titles such as: "A Sculptural Interplay" or "Effectively the Concern Manifests", in a tongue-in-cheek attempt to add weight to the images (Turpin et al. 2019).

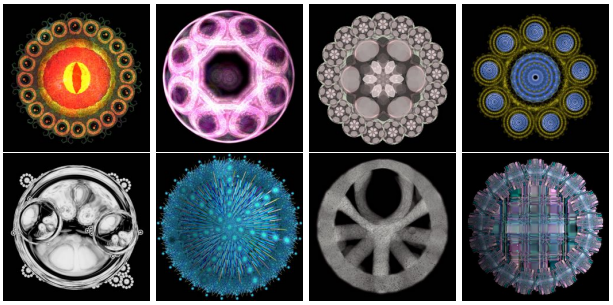


Figure 1: Sample images produced using the edit interface.

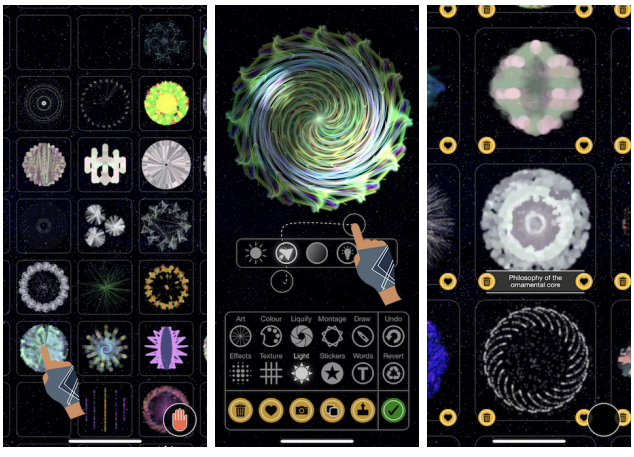


Figure 2: Art Done Quick screenshots (iPhone version).

Clustering

An important technique to help with the management of large numbers of images on the sheet interface, is the automatic clustering of images. There are numerous features of images that could be used for this, e.g., liked/not liked or based on the construction technique. However, the most sensible approach was to cluster together images in terms of how they look, taking into account shape, texture and colour. To do this, we based the clustering on the features generated by a headless application of the ResNet neural network mentioned above for image classification when captioning images. With these features, initial K-means++ clustering (Arthur and Vassilvitskii 2007) of images into (at most) 17 clusters appeared very successful, as the clusters have high internal visual coherence, but were sufficiently different to other clusters. However, the method didn't scale well, and users had to wait nearly six minutes (on an iPad Pro) in the worst case scenario, namely clustering 1,000 images. Given this, even a parallel clustering approach would still be far too slow for practical usage.

Headless applications of ResNet to an image result in a vector of 2,048 floating point numbers in the range 0 to 1. We call these numbers *ResNet features*. On inspecting the range of each feature, we noticed that many barely varied at all when applied to 10,000 randomly generated Art Done Quick images. We recorded the range of each feature over the 10,000 images, and ordered the features in decreasing range size. By restricting the features to just those with the highest ranges, we performed a kind of principal component extraction, so we can compare clustering times as the number of features used decreases. As shown in fig. 3, the time to cluster 1,000 images into 17 clusters reduces from nearly 6 minutes using all 2,048 features to around 2 seconds when the top 16 (in terms of range size) features were used.

Surprisingly, we found that using only the top 26 features produced clusterings which seemed just as high quality as those produced using all 2,048 features. Note that this is helped by the normalisation of the features to occupy the full range from 0 to 1, achieved using the feature ranges recorded over the 10,000 images. Note also that the distance measure for two features vectors used in the clustering calculates the average absolute distance between pairs of values in the same position in the two vectors. We call this the *ResNet distance* and use the function d to denote it. We settled on using 26 features so that each one could be assigned a different letter of the alphabet, which may help in rationalising how ResNet analyses images in future work.

The clustering takes at most 2 seconds on an iPad Pro and 3 seconds on an iPhoneX, and depends on the number of images to cluster. This delay is hidden from users in two ways. Firstly, as soon as the user taps the menu button, a new clustering is generated in a high-priority background thread, whether it is going to be used or not. As it takes users at least 1 second after this to navigate the menu and initiate a clustering, this hides at least half the processing. Secondly, an animation of the set of images disappearing takes around 1 second and occupies the user's attention while the clustering is finalised. Note that the headless application of ResNet is done at the time of an image's creation or alteration, not directly before the clustering, which saves time.

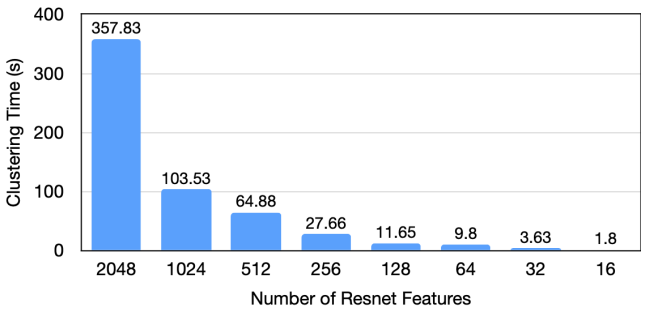


Figure 3: K-means++ clustering times for 1,000 images, varied by number of ResNet features used.

In clusterings of n images, the number of clusters is chosen proportionally to n, as $min(17, ceil(n/10))$, so at most 17 clusters appear. These are arranged in circles nestling within a large outer circle, as portrayed in figure 4(a). We are working on a prototype captioning system which will be able to lyrically label each cluster. Currently it uses information about major colours present in a cluster, and appends an arbitrary noun taken from an appropriate but generic set, including *bubbles*, *jewels*, *rings*, etc. For instance, in figure 4(b), two cluster captions, *Japanese laurel gemstones* and *snowy mint lights* are shown. Future versions will employ ResNet analyses to assign captions more appropriately, using shape and texture, as well as colour information.

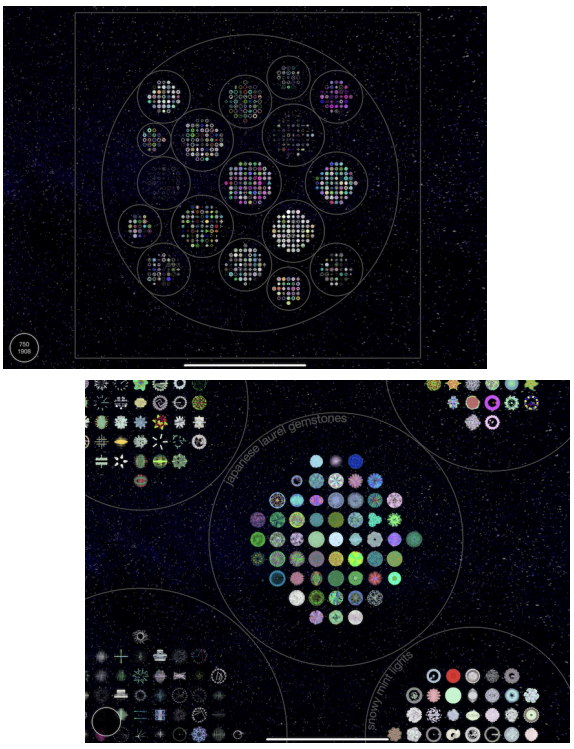


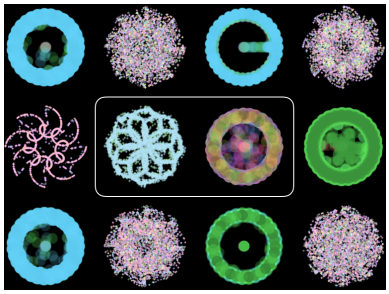
Figure 4: (a) Screenshot (iPad Pro version) of the clustering of 750 images. (b) Close up screenshot of captioned clusters.

Crossover

The sheet interface in Art Done Quick affords a simple and effective way of searching the space of raw-material images in an evolutionary fashion: (i) double tapping image I produces 8 images in neighbouring cells, with each image being a variation of I, and (ii) a long press picks up an image, I, enabling it to be dragged and dropped onto another image, J, producing 10 offspring of I and J in neighbouring cells. In both cases, existing images in any cell are moved away before the new ones are added. The mutation of one image into a set of new images has been long-solved in the sense of producing satisfying variations of the image. By *satisfying*, we mean that each variation look somewhat like the original and the other variations, but not too much like them. To expand upon this: if the variations are too similar to the original, the user may feel like he/she is not making enough progress with their choices, but if a variation is too dissimilar, the user may feel as if they have been served up random images, and have rather wasted their time.

To achieve a satisfying variation of image I, Art Done Quick mutates the `init` and `update` genes in the chromosome for I, with a mutation rate chosen randomly as either 1 or 2, leaving the `render` gene and `postprocess` gene untouched. On average, this produces images which are neither too close nor too far away from I, with mutation rate 2 naturally making images further from I than those with rate 1. A low mutation rate was chosen because we found that altering `update` was quite disruptive, producing images quite far from the original. For this reason, and as `render` and `postprocess` are available for users to edit directly, no mutation of these genes is performed when variations are generated, although this may happen in future work.

While the crossover of two images to produce offspring resembling them both holds the promise of an entertaining and interesting addition to Art Done Quick, this functionality has been added only recently, largely because the process was quite unsatisfying for users. This was because the offspring routinely looked either too similar to one of the parents or completely different to both parents, and this behaviour seemed invariant to changes in crossover methods. Moreover, the offspring often looked too similar to each other, i.e., there wasn't enough variety in the set produced. As an example, consider the following images:



The ten images around the outside were produced as offspring to the 2 images on the inside. The results are unsatisfying, as some of the offspring look too similar to each other, so there is redundancy, and all the images look too similar to one parent while not resembling the other enough.

Measuring the Value of Crossover Schemes

A crossover scheme consists of a method for combining the genetic material of two images into a third image in such a way that 10 different, satisfying, offspring images are produced when the genomes are used in the generative process. Note that image (phenome) based crossover would in theory be possible, using a range of image compositing techniques (Porter and Duff 1984). However, this is ruled out, as Art Done Quick re-generates thumbnail images at higher resolution, which can take longer than a second. If it also had to re-generate two parent images, and possibly grandparent and great-grandparent images, this would undoubtedly ruin the casual experience with long waiting times.

In order to find a good crossover scheme, we stipulated some requirements as follows:

[R1] A set of offspring images must be generated in one shot, i.e., the scheme produces exactly 10 images in one go.

[R2] Individual offspring images should be neither too visually similar nor too to dissimilar to either parent.

[R3] The set of offspring images produced must look reasonably similar and reasonably dissimilar to each other.

[R4] An individual offspring image should not take any longer to generate than a random raw-material image.

[R5] An offspring image should not occupy more memory than a random raw-material image.

Requirement R1 ensures that users are not waiting a long time for the offspring to be produced, as generating 10 offspring already takes around 1 second (iPad Pro) and 1.5 seconds (iPhone X). This rules out producing more images than needed and choosing the best from them. Requirements R2 and R3 are motivated by the discussion above about the results of crossover being unsatisfying. Requirements R4 and R5 are to make sure that crossover doesn't ultimately grind the app to a halt or overly reduce the number of images allowed on the sheet, as both the fluid/fast nature of user interaction with the app and the ability to have hundreds of images on-screen are *Prime Directives* (core aspects) of the experience users have with Art Done Quick.

In order to test different schemes for crossover with respect to requirements R2 and R3, we devised methods to capture (a) how good an offspring is with respect to looking somewhat (but not too much) like its two parents, and (b) how much a set of offspring look like each other. As a preliminary to this, we determined a *sweet spot* ResNet distance by generating 1,000 images randomly with Art Done Quick, then averaging the ResNet distance between each one and a mutation of it (produced with mutation rate 1). The average was 0.08, and given that the mutation method in Art Done Quick generally gives satisfying variations, this sweet spot distance captures the notion of a variation image being close enough to the original, but not too close.

Given a crossover scheme, S, applied to two parents P_1 and P_2, producing an offspring, C_1, we can calculate a **crossover score** as follows. Firstly, the ResNet distances between both parents and the offspring, $d_1 = d(P_1, C_1)$ and $d_2 = d(P_2, C_2)$, are calculated. Next, the absolute distance

	Crossover scheme						
	Baseline		Single Chromosome			Multiple Chromosome	
	Random	Copy	Init/Update	XY/RGB	Particles/Render	Equal Weight	Swampfix
No mutation	0.061	0.119	0.36	0.345	0.361	0.497	0.514
Single mutation	-	0.327	0.335	0.333	0.339	0.461	0.467
Double mutation	-	0.346	0.323	0.315	0.323	0.412	0.415

Table 1: Comparison of the crossover scores (to 3 d.p.) for 19 different schemes, averaged over 500 tests.

of d_1 from the sweet spot 0.08 is calculated, as is the absolute distance of d_2 from 0.08. The average of these two values captures how close a child is to the sweet spot distance from both its parents. To turn this into a measure with top value 1 representing a perfect offspring, the overall calculation of $crossover(P_1, P_2, C_1)$ is as follows:

$$1 - 10\left(\frac{|d(P_1, C_1) - 0.08| + |d(P_2, C_1) - 0.08|}{2}\right)$$

Note that if C_1 is exactly at the sweet spot ResNet distance from both P_1 and P_2, $crossover(P_1, P_2, C_1) = 1$. Note also that the multiplication by 10 helps normalise this measure over the range 0 to 1. As we shall see in an experiment described below, randomly generated (i.e., with no reference to either parent) offspring rarely have ResNet distance from either parent more than 0.18, so the absolute distance from the sweet spot (0.8) is rarely more than 0.1. Hence, dividing by 0.1 (or equivalently multiplying by 10) normalises the average of the two sweet spot distances. While this means that some crossover scores can be below zero, this is only for particularly poor offspring, and the normalisation helps with understanding the value of a scheme.

Given n offspring $C = \{C_1, \ldots, C_n\}$ resulting from a crossover operation, we calculate the **sibling score** for C as:

$$1 - \left(\frac{10}{{}^nC_2}\right)\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|d(C_i, C_j) - 0.08|\right)$$

This averages the distance from the sweet spot of the ResNet distance for each distinct pair of offspring, and normalises it as above. Again, values for this tend to be in the range 0 to 1, but can be negative in very bad cases. This measures how good a set of offspring are with respect to requirement R3 above, i.e., whether there is enough (but not too much) visual variety in the offspring arising from a crossover operation.

To try and improve the quality of the crossover in Art Done Quick, we compared various schemes in terms of the crossover score for individual offspring. Two obvious baselines to compare against here are (a) the scheme which outputs a **random** image as an offspring and (b) the scheme which outputs a **copy** of the first parent. Recalling that the genome of an image contains a single chromosome consisting of four genes: `init`, `update`, `render` and `postprocess`, we concentrated on crossover of raw-material images, i.e., with no post processing. We implemented and tested the following 3 schemes for the chromosonal crossover of parents P_1 and P_2 into offspring C_1:

- **Init/Update**: C_1 inherits the `init` and `render` genes from P_1, but the `update` gene from P_2.
- **XY/RGB**: C_1 inherits the x and y bases from P_1 for both `init` and `update` genes, along with the `render` gene, but the r, g and b bases from P_2.
- **Particles/Render**: C_1 inherits the `init` and `update` genes from P_1, but the `render` gene from P_2.

To compare the quality of the output for these schemes, we generated 500 pairs of images randomly, and produced a single offspring for each of the random, copy, init/update, xy/rgb and particles/render schemes. In addition, for the non-random schemes, we also generated two mutations of the offspring, one with mutation rate 1 (single) and one with mutation rate 2 (double). The results are given in table 1. We see that, as expected due to the normalisation, the worst case of generating a random offspring produces an average crossover score near to zero, namely 0.061. The scheme which takes a copy of P_1 as the offspring suffers from the offspring being not related visually to P_2, and from it being too visually similar to P_1, hence neither ResNet distances will be close to the sweet spot, and it scores only 0.119. Mutating the copy improves matters, because the offspring no longer looks as similar to P_1 and the scores for mutation rates 1 and 2 are 0.327 and 0.346 respectively.

We also see that, unfortunately, none of the three crossover schemes above significantly outperforms the copy + mutation schemes, with the highest score being 0.361, for the particles/render scheme. These results re-affirmed our experience that crossing over material at genome level into offspring produces an unsatisfying experience in terms of the images produced. Further evidence for the disappointing results are given in the left hand column of figure 6, where the particles/render scheme has been used for crossover.

Multi-Chromosome Crossover

We achieved a breakthrough with the crossover in Art Done Quick by adhering to the fun-first methodology described in (Colton et al. 2020a), based on the specification of casual creators given in (Compton and Mateas 2015). In one sense, crossover is redundant in Art Done Quick, because the space of images can be efficiently and enjoyably searched with just the random generation and mutation functionalities. However, given the pleasing nature of dragging one image onto another and seeing what results, we decided to implement crossover, but concentrated on how to increase the fun in this, rather than on how to produce good images. In particular, we returned to a previously rejected idea for crossover,

which we originally felt would degrade the quality too much, as the images produced would become rather complicated and disordered. Taking a more positive view, however, we determined that this enables users to produce *messy* images, and – as any child will tell you – getting messy is fun, fitting well into the fun-first design approach.

In summary, with the new crossover approach, each offspring inherits all the genetic material of both parents, with each having half the control over the way in which the offspring image is generated. Recall that raw-material images are produced by a set of particles changing position and colour over time. A natural way of splitting the responsibility, therefore, is for the chromosome of parent P_1 to control the even-numbered particles, and for parent P_2 to control the odd-numbered particles. In this way, the shapes rendered for the child consist of half of those of each parent, interleaved.

This new approach required a change in representation, with genomes now containing multiple chromosomes and an additional specification of which chromosome controls which particles. With the new representation, when a child inherits material, each chromosome from both parents is assigned a set of particles to control when the child image is generated. This can lead to a large number of chromosomes in certain genomes, so a cap of 32 has been set, and whenever the number of chromosomes to be inherited exceeds this figure, some are randomly chosen to be ignored.

We have found that, as expected, the images produced do get increasingly disorganised as offspring inherit more and more chromosomes, and this does feel analogous to getting messy with traditional art materials. Figure 7 portrays three family trees with 16, 8 and 8 images, respectively, randomly generated and crossed over until there is a single descendent inheriting the genetic material from all of them, with these images shown in more detail on the right of the figure. On visual inspection, the offspring largely look like both parents, but are somewhat different, and this seems much improved over the previous schemes. The images are indeed more disorganised than the randomly generated or mutated raw-material images, but we've found that with careful choices of what to crossover in the app, it is possible to produce aesthetically pleasing (albeit subjectively) images, and this has opened up a wide range of imagery not previously available, such as those presented in figure 5.

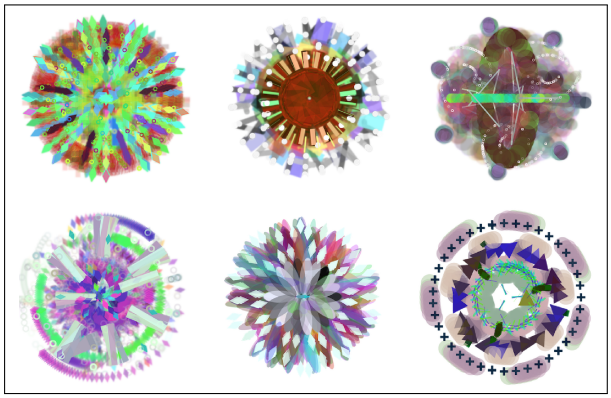


Figure 5: Images made using multi-chromosome crossover.

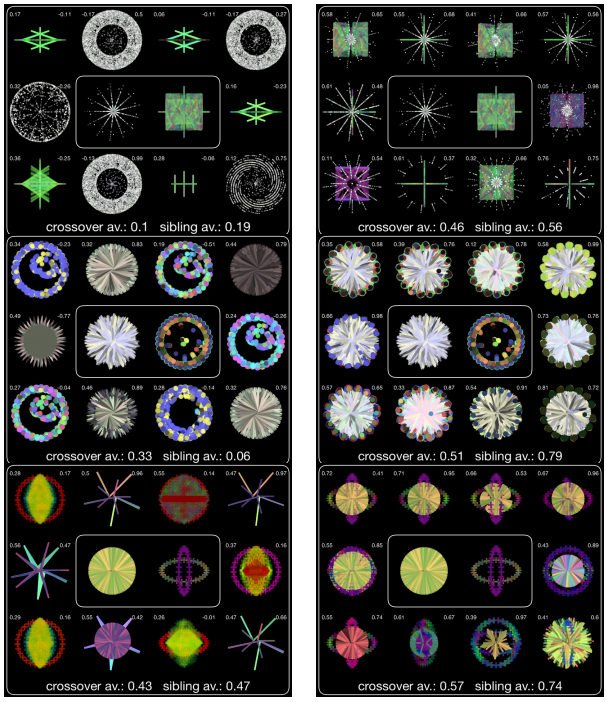


Figure 6: Left column: crossover with the particles/render single chromosome method. Right column: crossover with the swampfix multi-chromosome method. The same parent images are used on both sides of each row, for comparison. Average crossover and sibling scores are given.

One issue with the new crossover approach is *swamping*, where an offspring has little resemblance to one of its parents. For instance, under an **equal weighting** crossover scheme, if a parent image generated by rendering a small number of small shapes is crossed with one generated by rendering a large number of large shapes, the shapes from the latter will dominate in the offspring rendering, which produces disappointing results. To cater for this, we implemented a **swampfix** scheme where the two parent images are analysed in advance to calculate the proportion of background to foreground (shape) material. The number of offspring particles assigned to the chromosomes of each parent is calculated with respect to these proportions, so that parents with a higher background proportion are given up to ten times more particles in the offspring than a parent with a higher foreground proportion. We've found that this often brings out the characteristics of a swamped parent in the offspring, but there are still some issues, which we will address in future work. Note that swampfix has similarities to probabilistic crossover techniques, such as those described in (Ortiz-Boyer, Hervas-Martinez, and Garcia-Pedrajas 2005).

To verify our perception that the images produced by the multi-chromosome crossover are more satisfying than those produced by the older schemes, we generated 500 pairs of images randomly and calculated the crossover score for the equal weighting and swampfix schemes. We also tried schemes where either a single or both parents were mutated (with mutation rate 1) before they were crossed over, and likewise recorded the crossover score. Table 1 presents

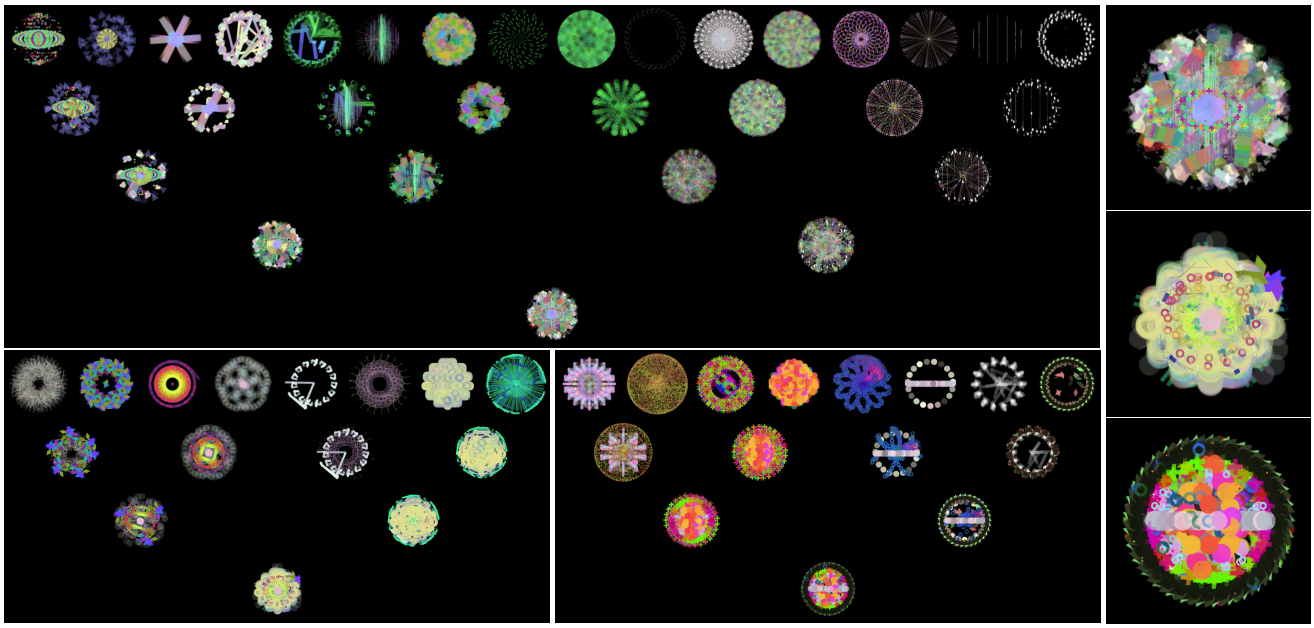


Figure 7: Family trees generated via the repeated use of multi-chromosome crossover, given with the final image for each tree. With the exception of the top row image, each image is the offspring of the two directly above and to the left and right of it. On the right are the final descendant images of the three trees, shown in greater detail.

the results from the experiment, where the same 500 image pairs were crossed over via 19 different schemes, including, as mentioned above, the baseline and older schemes and mutation-variants thereof. We see that the equal weight and swampfix schemes produce much higher crossover scores (of 0.497 and 0.514 respectively) than the older schemes, the best of which scored only 0.361. This matched our perception that the crossover in Art Done Quick is now quite satisfying: producing messy, varied, offspring images which look suitably like their parents. These results also support the use of the swampfix method, as it scores higher than the equal weighting version of multi-chromosome crossover.

We also note from table 1 that the mutated versions of the equal weight and swampfix schemes, while scoring less than the non-mutated versions, still score fairly high. Recall that Art Done Quick needs to generate 10 images to fill neighbouring cells on the sheet when a user initiates crossover. Requirement R3 states that these images should also be suitably different to each other, and the sibling measure described above can be used to evaluate this. We experimented with two swampfix schemes for generating a set of 10 images: (a) two images produced through crossover, with the second one using reversed parents, and eight more images produced this way, but with one parent pre-mutated, and (b) two images produced through crossover, four with one parent pre-mutated and four with both parents pre-mutated.

To test how these score with respect to the sibling measure, we generated 500 image pairs randomly, and produced 10 offspring with these two schemes. For comparison, for the same set of image pairs, we also generated 10 offspring using two particles/render single-chromosome crossover schemes. In the first scheme, the first two images were produced with standard crossover, reversing the two parents for the second one, then 8 images were produced with these crossovers followed by a mutation with rate 1. The second scheme was the same, but with four of the eight mutated images produced using mutation rate 2. The average sibling scores for the crossover schemes were as follows:

	Scheme	
	Particles/Render	Swampfix
Single Mutation	0.308	0.682
Double Mutation	0.033	0.648

We see that swampfix scores quite highly, and more than double the particles/render crossover scheme. Some sample crossover operations from the single mutation schemes are given in figure 6, where we see that the results from swampfix are clearly superior to those from the particles/render scheme. This is borne out (subjectively) visually, and in the crossover/sibling scores, which are included in figure 6.

As a final consideration, we return to requirements R4 and R5. Firstly, the time taken to generate an offspring image is roughly the same as with single-chromosome raw-material generation, as equal numbers of shapes are rendered. However, an unwanted side effect of the multi-chromosome crossover is that the images are more complex and hence less amenable to compression. Art Done Quick generates images for the sheet at a resolution of 500×500 pixels, so that there is no loss of fidelity when regenerating at a scale of 2000×2000 pixels. However, to enable it to show so many images on the sheet, it stores the image in memory at 180×180 pixels and applies JPEG compression with quality 0.7. These values have been derived to produce images with the lowest memory footprint for an acceptable loss of image quality, and this approach enables 1,000 images to be shown on the sheet at one time (on an iPad Pro).

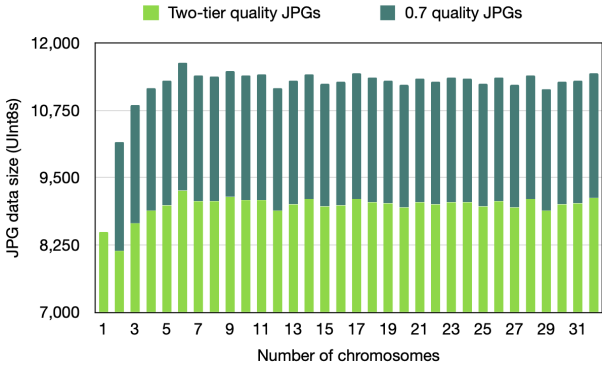


Figure 8: JPEG image size varying by no. of chromosomes.

We noticed that swampfix crossover images had bigger memory footprints than single-chromosome ones. On investigation, we found this was because the JPEG compression was not as effective on the more complex images. To further investigate, we generated 32 sets of 100 images ranging over 1 to 32 chromosomes, and calculated the number of UInt8 data points that were required to store them in memory, after scaling and JPEG compression as above. The results are given in figure 8 as the top (darker) bars in the chart. We see that there is a steep increase in storage size from 1 to 2 chromosomes, but that the data size does not increase after five chromosomes. This prompted us to introduce a two-tier compression scheme, whereby images generated via more than one chromosome are compressed at 0.6 quality, rather than 0.7 for single-chromosome images. On visual inspection, we found that the more complex multi-chromosome images were able to be compressed with only 0.6 quality without unacceptable image degradation. Moreover, as can be seen in the bottom (lighter) bars of the chart in figure 8, this approach brings down the storage size substantially, to a level similar to single chromosome images.

Conclusions and Future Work

While we have presented only certain developments in a particular project, there are some points which might help casual creator design in general. In particular, the fun-first design methodology is a powerful driving force which provides opportunities for implementations and research questions that would probably not be unearthed if a generative system was designed, say, for professional art production. Moreover, this methodology can help in solving problems that arise because of such perspectives. In casual creator design, artefact quality has to be tensioned against user enjoyment, with the latter often taking precedence, e.g., producing messy images is fun even though the images may seem lower quality. Also, while it is possible to use subjective evaluation of difficult notions such as how satisfying a crossover operator is, there are benefits to formalising a measure for this, so that different approaches can be objectively compared and contrasted in large-scale experiments.

The image generation process in Art Done Quick was originally developed as part of The Painting Fool project (Colton et al. 2015). To compare human and AI creativity, and to study people's perceptions thereof, we plan to re-unite the two projects with The Painting Fool becoming a user of Art Done Quick, as per (Colton et al. 2020b). Art Done Quick is being developed as both a research platform and a casual creator app for commercial release. We are nearing the end of stage one of the development, which will result in the first fully-featured version of the app, and we will soon undertake user testing. We also hope that the app will serve as a platform for showcasing generative and allied technologies: it already employs generative visual art techniques, generative text techniques and machine vision. Moreover, we have already implemented, and are currently fine-tuning neural style transfer techniques (Gatys, Ecker and Bethge 2016) to add fun custom-generated textures to images. We also plan to experiment with vision-informed generative audio design, so user interaction with an image is accompanied by a subtly different set of sound effects. In this way, we hope to attract mainstream app designers to turn to computational creativity research when making casual creators.

Acknowledgements

Many thanks to the anonymous reviewers for their insightful feedback, in particular the notion of a casual creator "Prime Directive". Many thanks also to Jon McCormack, Simon Lucas and Penousal Machado for each insisting that crossover would be a fun addition to Art Done Quick.

References

Arthur, D., & Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *18th Ann. ACM-SIAM Symp. Disc. Algorithms*.

Colton, S.; Halskov, J.; Ventura, D.; Gouldstone, I.; Cook, M.; and Perez Férrer, B. 2015. The Painting Fool Sees! New Projects with the Automated Painter. In *Proceedings of ICCC*.

Colton, S.; McCormack, J.; Berns, S.; Petrovskaya, E.; and Cook, M. 2020a. Adapting and enhancing evolutionary art for casual creation. In *Proceedings of the EvoMusArt Conference*.

Colton, S.; McCormack, J.; Cook, M.; and Berns, S. 2020b. Creativity theatre for demonstrable computational creativity. *ICCC*.

Colton, S.; Berns, S.; and Pérez-Ferrer, B. 2020. First experiments in the automatic generation of pseudo-profound, pseudo-bullshit image titles. In *Proc. AISB symposium on computational creativity*.

Compton, K., and Mateas, M. 2015. Casual creators. *Proc. ICCC*.

Compton, K. 2019. *Casual Creators: AI Supported Creativity for Casual Users*. Ph.D. Diss., University of California, Santa Cruz.

Gatys, L.; Ecker, A.; and Bethge, M. 2016. Image style transfer using convolutional neural networks. In *Proc. CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Ortiz-Boyer, D.; Hervas-Martinez; and Garcia-Pedrajas, N. 2005. CIXL2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research* 24.

Pennycook, G.; Cheyne, J.; Barr, N.; Koehler, D.; and Fugelsang, J. 2015. On the reception and detection of pseudo-profound bullshit. *Judgment and Decision Making* 10(6).

Porter, T., and Duff, T. 1984. Compositing digital images. *Computer Graphics* 18(3).

Romero, J., and Machado, P., eds. 2007. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Springer.

Rule, A., and Levine, D. 2012. *Int. Art English*. Triple Canopy.

Turpin, M.; Walker, A.; Kara-Yakoubian, M.; Gabert, N.; Fugelsang, J.; and Stolz, J. 2019. Bullshit makes the art grow profounder. *Judgment and Decision Making* 14(6).