Linux GIMPのプラグイン:SICで目指す芸術と科学の融合
-数行のプログラムの変更がもたらす個性的なCG表現の世界-

笠尾敦司
東京工芸大学　芸術学部　デザイン学科 kasao@dsn.t-kougei.ac.jp

1. はじめに

芸術と科学の融合は色々な局面で望まれているが、真の融合を達成するには、両者が互いの資質をオープンにする、つまり、人間レベルでのオープンソース化が必要である。
芸術家と科学者が協力する機会は日増しに増えている。友好的な協力関係が是非とも必要である。しかし、芸術家と科学者が協力するというのは言葉でいうほど簡単ではなく、両者が近づくことで逆に両者のいがみ合いを誘発してしまうことさえある。そのため、それを避けるために役割をはっきりと分担してそのようなことが始めから起こらないように企画を立てることが最近では多くなってきているように思われる。確かに、一つの目的を達成するためには効果的な方法であるが、この方法では、道具を作る人とそれを使って表現する人に分かれてしまいがちになり、芸術家と科学者が協力はしているが、真の融合とはほど遠いものになってしまう。
予定調和的に作品が作られるのではなく、始める前には予想できないような新しい価値や表現が生まれるような本来あるべき融合した協力関係を生み出すためにはどのようにしていくべきなのかを考える必要がある。
一方、我々は、プログラマーがプログラミングするという行為の結果として2次元のCGアートを作り出すことのできるSIC (Synergistic Image Creator) を研究開発している。このSICはLinuxのGIMPのプラグイン集として作られており、現在、ネットワーク上で出会った面識のない人同士が、SICのオープンソース化を目指して、協力しながらソースコードの整備を行っている。
このSICは画家が作品を作り出す際の一連の処理の流れ、つまり、絵を描く対象から特徴を抽出し、その特徴を取捨選択あるいは強調して、自分なりの表現として作品を作り出すという過程をステップごとにプラグイン化してあるところに特徴がある。また、この一連の処理で用いられた処理パラメータは作り出された画像ファイルのコメント欄に書き込まれるため、他の人が作り出したCG作品で気に入った画風があれば、そのパラメータを一部参考にして、さらに自分なりの処理を行うことができる。今までのCG作品はたとえもと絵が写真であっても、処理が残っていないので、どのように制作したかは分からなくなっているが、SICでは作り出されたCG作品と処理パラメータの対で一つの作品と考えているため、制作過程が分かるようになっている。
つまり、作り出されたCGもオープンソース的なのである。しかし、SICの本当の特徴は単にパラメータを変化させていろいろな画風を作り出すというだけではなく、プラグインの数行を変更することで真にオリジナルな処理を加えていけるということにある。アルゴリズムそのものを変更することで表現を行うため、このこのようにして作成されたCGをアルゴリズミックアートと呼んでいる。
もちろん、SICが写真画像を絵画風に味付けする単なる画像処理フィルターと同じ程度の能力しか持たないものであれば、魅力はないので、SICは上記の特徴の他に、多様でかつ高品質な表現を生み出す能力、元画像が劣化してたり小さくてもポスターサイズの大きな作品を作り出す能力、なども併せ持っていることも大切な特徴になっている。
「プログラミングするという行為の結果としてCG画像での独自の表現が可能になる」という特徴を生かして、SICに芸術と科学技術の相互理解の架け橋とする役割を担わせたいというのが、我々の最も大きな目標である。
本論文の構成について以下に述べる。第2章において、芸術と科学の融合が困難な理由を考察し、第3章で、SICを利用して作成するアルゴリズミックアートが第2章で述べた芸術と科学の融合に有効に働くかどうかを検討した。次に、第4章では第3章で述べた特徴をSICの処理過程に従ってSICを構成するプラグインを説明する。第5章では作品例とそれに対応するプラグイン変更の実例を示す。第6章でアルゴリズミックアートの特徴を十分に生かせるように作れらたホームページを中心としたSynergistic Art Projectを紹介し、最後に第7章でまとめをおこなう。本論文を読むことで、実際に科学に携わる人も芸術に携わる人も、アルゴリズムアートに取り組むことができるように、記述には具体性を持たせるよう心がけた。

2. 科学と芸術の融合を阻害する要因

多かれ少なかれ、芸術的思考と科学的思考が一人の人間の

二つの側面として現れる必要があると筆者らは考えている。もちろん、それぞれの専門性をなくすということではなく、芸術と科学の境を感じずに自由に自分の専門性を生かして活動できる人材が必要であるということである。

しかし考えてみると、人間は本来的に芸術的でもあり科学的でもあるはずのものである。つまり、あるものに感じてその感動を表現したいという気持ち、そして、世界に満ちている不思議や謎を解明したいという気持ち、誰しもが子どもの時には持っていたはずである。これを大人になるまでの長い間に忘れてしまったのであろうか。幼稚園の頃にお絵かきが大好きという子は多いが、小学校の高学年になる頃にはその子どもの大部分はすっかり絵を描くことが嫌いになっている。またこれは、数学などの勉強に関しても同じで、低学年では先生の質問に対し一斉に手を挙げるが、高学年になると数学嫌いも顕著に増えてしまうことは周知の事実である。だが、科学に携わる者でも楽器や絵画で自己を表現する人は少なくはない。では、何故であろうか。

言葉通り、芸術は表現するもので、科学は極めるものであり、科学者と芸術家にもそのままその傾向が現れている様に思える。つまり、芸術に携わるものは自分の主観を表現し、それによって多くの人の共感を得えようとし、科学者は客観的な法則を見つけたり、客観的な目標値を達成することで自身の満足を得ようとする傾向がある。従って、科学に携わっている者からすると自分の主観を表現するために科学技術を利用するということは、今までの態度とは180度異なることになる。単に表現に慣れていないということではなく、明確な目標のない自己の主観を表現するために科学的な手段を使うことに慣れていないのである。

SIGGRAPH等で発表されるNon-Photo Realistic CGに関する発表も印象派の絵画に似せるための筆触づくりであったり、不自然さの少ないpen & ink expression に関する論文ばかりである。Non-Photo Realistic CGのプログラミングであっても客観的な価値を実現するためになされているのである。プログラマーが自己を表現する作品作りのためにNon-Photo Realistic CGのプログラミングをした例を私は知らない。

逆に芸術に携わる者からすると科学技術も自分の表現を実現するための道具と見なすことができるはずである。しかし、彼らには十分に技術を使いこなすだけの技能に不足していることが多い。このような状況下において、科学に携わるものと芸術に携わる者が一つの作品を作るために集められると、極端な場合、芸術家の設定した目標を達成するために技術を積み上げる道具の一部として科学者が働くことになってしまう。

ところで、ヨーロッパでは子どものうちからメガデモ[1]に見られるように自己の表現のためにプログラミングをするという伝統がある。このメガデモは著作権上は多くの問題を孕んでいるが、逆に全てがオープンになっているところに教育的な価値は高いものと考えられる。メガデモがオープンであるという意味には、プログラムそのものがオープンであるというだけではなく、作り上げられた作品も勝手に利用してしまうという意味で、作品そのものもオープンといえる。法律的には幸いなことかもしれないが、日本にはこのような伝統がなかったといって良いだろう。

このような状況を打開するためには、まずは人間は芸術家ではなくても、根本的に誰でもが表現者であることを再確認することが大切で、次に大切なのは、科学に携わる者も、自己の表現のためにこだわりなくプログラミングを行えるようになることである。そして、そのことによって、企画段階から芸術家と共に積極的に作品作りに関われるようになり、自ら独自の表現を作り出す様にも自然に成れるはずである。

また、一般に作品作りはまねをすることから始めるのが基本であることをからも、制作するためのプログラムやそれを用いて作られた作品もできる限りオープンにする環境が必要であろう。

最後に、芸術サイドから考えてみる。もしも、簡単で且つ少しのプログラミングの変更でオリジナルな作品が作れるのであれば、プログラミングの技術がほとんどなくても作品を作り出すことができる。芸術家がそのような道具を使うことのできる環境を手に入れれば、実際にそれを用いて作品を作るかどうかは別にしても、それを用いることで、科学的な思考やプログラミングについて学ぶことができるようになる。つまり、芸術サイドからも歩み寄る契機になるに違いない。

3. アルゴリズミックアート

以上のように考察してきた条件を筆者の提唱するアルゴリズミックアートに当てはめて考えてみる。アルゴリズミックアートとは文字通り、筆を走らせる代わりにアルゴリズムを変えながら試行錯誤することで、自己の表現を行うことまたはその行為によって作られた表現作品のことである。このアルゴリズミックアートを作り出すソフトウエアーとして研究開発されたのが、Synergistic Image Creator(SIC[2)3)]) である。前章において、科学的な手段で主観的な作品を作り出す環境として大切なのはオープンな環境であると述べたが、SICはオープンソースであるLinuxのGIMPプラグインとして作成されている。また、SIC自体もオープンソースとして公開できように現在準備を進めているところであり、この論文が発行される頃には自由に使えるようになっているはずである。

SICはアルゴリズムアートとして作品を作り出すために変えられる部分と画像処理を行う基幹部分とに分けられる。

表1にSICを構成するプラグインをまとめた。この中の1-4GKReconstImageと1-5GKSubstColorの二つのプラグインが作品を作り出すために変えられる部分であり、他のプラグインはSICの基幹部分である。従って、アルゴリズミックアートを試み

る者は上の二つの一部をさまざまに変えて作品を作り、SIC開発に関わるコアメンバーはSICの基幹部分整備する。どちらも違うレベルでSICがオープンソースであることを有効に利用していることになる。
日本では子どもから大人になる過程のどこかの時点で何かしらの壁を自ら築くことで、科学者または芸術家になったはずである。従って、その壁を壊しオープンにすることが大切で、このようなオープンな力をSICは潜在的に持っているはずである。SICを用いることで芸術サイドからも歩み寄る契機になると前章で述べたが、それは人間が絵を描く際の情報の処理の流れに似た処理過程をSICが持っているということも理由の1つである。
これを説明するためにSICを用いたアルゴリズミックアートの制作課程を画家が絵を描く場合のものと比較してみることにする。
絵画は、「ある風景を見て、自分が強く感じた印象のもとになる特徴をつかみだし、その特徴を自分なりに加工することで作品を作る。そして、その個人的な印象を的確に表現するために自らの腕前を磨きながら絵を描く」一方、SICによるアルゴリズミックアートは「ある風景をデジタル信号として取り込んだら、そこから、まず特徴を抽出し、抽出された特徴を自分なりに加工することで作品を作り出す。つまり、個人的な印象を的確に表現するためにアルゴリズムを変えながら絵を創る」ということになる。つまり、我々のアルゴリズミックアートとは、画家が絵筆を使いこなす創意工夫をソフトウエアーつまりアルゴリズムの変更によって表現としてのCG作品を作り出そうというものなのである。
芸術と科学の融合を目指すので、このアルゴリズミックアートを制作可能とするソフトウエアは芸術家にも科学者にも使いやすいものであるべきである。そのため前章で考察したことをもとに、このソフトウエアに必要と考えられる特徴を以下に整理してみた。
芸術家サイドから
特徴1. 制作過程の流れが直感的(絵画の制作と近い)なものであること
特徴2. アルゴリズムの変更が簡単であること

科学技術者サイドから
特徴3. アルゴリズムの変更とその効果がある程度予測しやすいものであること
特徴4. 制作に関する他の人の知見をたやすく取り込むことができること

そして、SICは以上の特徴を基本的に備えている。また、このSICを用いたオープンCGの活動もSICのホームページを中心に始められたところである。

1-GKlab
rgb画像を人間の視覚が捕らえる色の情報に近い色空間に変換する。

1 1-1GKDirec
画像のなかから方向性を見つけだすために、ロッド(直線)を見つけだす。

2 1-2GKmeanAlgo
画像を解析するための最小単位の領域(筆触)を作り出す。

3 1-3GKAddPrincipal
解析結果を基に筆触の形をベクトル情報に直す。

4 1-4GKIMerge
筆触を人間の視覚を参考にしなが統合する。

5 3-GKIAnalyse
ここでは、統合された筆触を一つの単位として明るさの変化を与えたり、最小単位の領域の色を変更したりという操作をする。

6 1-4GKReconstimage
筆触から作られたベクトル情報をもとに筆触を好みの形にする。

7 1-5GKSubstColor
作られた筆触に3-GKIAnalyseで変更した色を塗る。

表1 SICを構成しているプラグイン

4. Synergistic Image Creator

アルゴリズミックアートの中核をになうSICの二つのプラグインについては後に述べるが、その前にSICの概略について述べる。
SICはGIMPを立ち上げる際にSICを読みこませることで利用できるようになる。前章の特徴1で示したように、SICは人間の視覚処理を参考にして、プラグインが構成されているので、以下に表1に示したプラグインの中から主要な物を抜き出し、一般的な処理の手順に従って説明する。項目の番号と図1中の矢印についている番号は対応している
1. **1-1GKDirec**
$L^*a^*b^*$に変換した画像から絵を描く際の特徴となるエッジを線素として抽出し、それをエッジ特徴画像として保存しておく。
この辺までが網膜上の処理に対応すると見なせる。
2. **1-2GKmeanAlgo**
先の$L^*a^*b^*$画像と上で抽出された線素の情報を用いて細かな領域(筆触)に分割し、それぞれの筆触に番号をつけ、それぞれの中心値(center), 平均色(color), 方向(direc)をmapに整理する
3. **1-3GKAddPrincipal**
mapに整理された情報をもとに解析して、map_28 または、map_28_abstを作成する。
4. **1-4GKIMerge**

似た特徴をもった筆触をまとめて、大きなグループを作る。その結果として以下の3つのファイルが作られる。
map_28_merged, unifymap_28_merged, mapinfo

5. **3-GKAnalyze**

map, map_28, map_28_merged, unifymap_28_mergedなどの値を使って、明るさの補正を行ったり、色度を変化させるなどの変化を与えて、新たに、
map_28_abst, map_28_merged_abst, unifymap_28_merged_abst
または、
map_28_imp, map_28_merged_imp, unifymap_28_merged_imp
を作成する。

6. **1-4GKReconstImage**

上に示した3つのmapセットのどれかに始めに作成したmapを加えたものを用いて、さまざまな形の筆触を作り出す。また、ここで、作品の最終的な大きさを考えて拡大率を決定する。もとの画像が小さくてもポスターサイズまでジャギーのない拡大を行うことができるというのもSICの大きな特徴である。図1に示した省略可能なマップという枠に囲まれているマップは利用しなくても一応の処理は行える。つまり、4, 5の処理は作品の質を無視すれば省略可能と言うことである。

7. **1-5GKSubstColor**

6と同様に3つのマップセットのどれかを用いて、6で形が作られた筆触に色を付けて処理を終える。これも6と同様に、図1に示した省略可能なマップという枠に囲まれているマップは利用しなくても一応の処理は行える。

以上が処理の概略である。
このようにSICは人間が絵を描く際の情報処理に近い処理の流れを持っているので、強く印象を感じた特徴を抽出し、その特徴を強調あるいは無視することで作品化していくという一般的な作品制作のおおよその流れを実現することが可能である。また、図1に示したようにこれら一連の処理の途中の処理を抜かしたり、組み合わせを変えることで、処理のバリエーションはさらに多様になる。これが、SICの利用を飽きさせない特徴になっている。また、図1に示してたmap中の平均色情報colorと領域の方向情報direcは異なる画像から作制可能であることがSICの表現の幅を広げることになっている。図2にはdirecを計算するために敢えて強いテクスチャーを描き込んだ画像の例を示した。この画像から分かるように、作り出された作品 (図2-c) は図2-bに引かれた黒い線に沿って筆触が向きを変えていることがわかる。
図3に図1中で説明している作品A,B,C,Dのうち作品B,Cの典型例を示す。ここでは、作品としての完成度は無視して、それぞれの表現の特徴がはっきり分かるような筆触表現を選択した。写真とほとんど同じ彩色になる作品aより昨作品bは色がばらつくがその様子が図3-aより分かる、また、この処理では明るい部分と暗い部分が画面に散らばっていることも特徴である、この特徴は絵画に対して、一般に絵画が持っている特徴であり、写真とは違うと見る者に感じさせる重要な要素になっている。次にcになると1つ1つの筆触の方向がはっきりしており、その方向は周りの筆触の影響を受けていることが分かる。この特徴を使うと、晩年のゴッホの絵にあるような糸杉の表現などが可能になる。この他にも、これらの作品を見て分かるとおり、SICの特徴の1つとして作り出される筆触には大きな物と小さな物があり、その差がかなり大きいということが挙げられる。他の絵画を単にまねた処理の多くはだいたい同じ大きさに筆触が

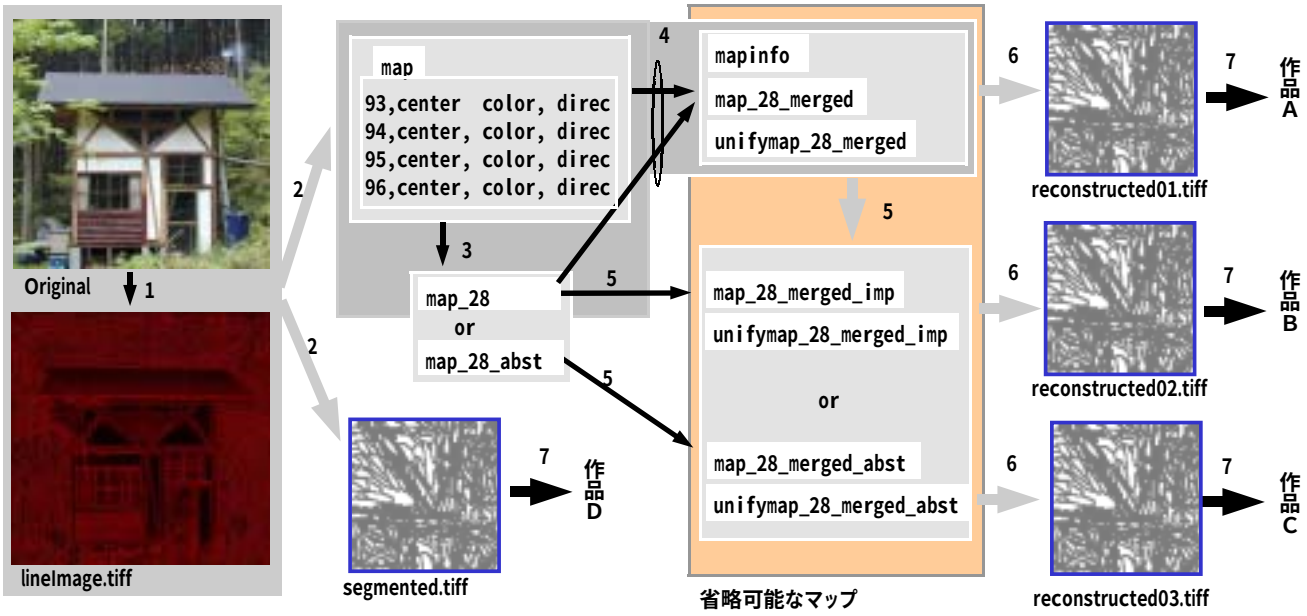

図1 SICの処理の流れ

a

b

c

図2　a 平均色情報を作り出すための画像、 b 方向情報を作り出すための画像、c 両者から作られたmapをもとに作品化した画像

a　図1の作品B

b　図1の作品C

図3　図1中に示した作品B,Cの例を示す

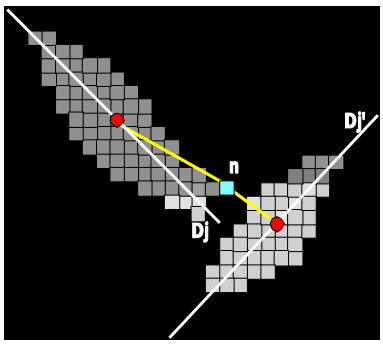

図4　筆触表現メカニズム

揃ってしまい、そのことが不自然さ感じさせることにつながっている。作品dは図には示していないが、ベクトル情報ではなく、単純にもとの画像の色だけを筆触を単位として変化させているので、ジャギーのない拡大はできないが、もとの画像がもっている緻密な情報を表現することができるため、使い方によっては有効な場合もある。

5. アルゴリズムと表現の関係

アルゴリズムアートとはいっても、アルゴリズムに自分なりアイディアを盛り込むのに多くの手間を要するのでは実際には利用しにくいものになってしまう。そこで、3章で述べた特徴2と特徴3つまり、アルゴリズムの更新が容易でかつ直感的であるという性質が大切となる。実際に作品を作制する際に、作品の表現を大きく変えることになるプラグインは6番目のGKReconstImageと7番のGKSubstColorである。

5.1 プラグイン GKReconstImage

GKReconstImageは図4に示すように画素nとその周辺にある筆触の中心(丸印)との距離を計算し、その距離がもっとも短い筆触にその画素が含まれるという規則で筆触を形作っている。画素nの周辺にどの筆触中心があるかはあらかじめ調べてその

```
if(mode==0)
{
direcf2=2./(1.+exp(7.*(0.30-direcf2)))-1.;
k4=1./(1.+exp(7.*(0.50-big_map[27][nab[mapmap][seg]])));
k6=pow(big_map[3][nab[mapmap][seg]],0.5);
deltaX=deltaX*(1.18+direcf2*k4)/k6;
deltaY=deltaY*(1.18+direcf2*k4)/k6;
}
```

図5　筆触表現ルーチンの例

結果を表にしてある。距離のみが筆触を決める条件なので、筆触中心と画素との距離の定義式を変化させることで色々な筆触が作りだされることになる。

プラグインGKReconstImageを選択した際に現れるウインドの最初のパラメータの値でどのような距離計算で筆触を作り出すかを指定しており、このパラメータはプログラム中ではmodeという変数が保持している。gk_reconst_image.cをエディタで開くと図5に示したようなif文を見つけることができる。

これは、もっとも基本的な mode=0 の場合の距離の定義式である。もしも、新しい距離の定義式を作り出すのであれば、新しい mode の番号 x をつけて、

if(mode==x)

という if 文を付け加えれば良い。この if 文以下を適宜変えることで色々な表現を作り出せることになる。

簡単に if 文以下のパラメータを説明するが、基本的に図 5 の 6 行目の deltaX と 7 行目の deltaY から計算される領域と注目画素との距離の平方 $deltaX^2+deltaY^2$ を領域の大きさや方向の情報に合わせて変えることで様々な表現が可能になる。

3 行目の右辺の direcf2 は図 4 の黄色い線 (画素から筆触の中心までを結んだ線) の方向を 0 ～ 1 の値で表現している。direcf2=0 のときに、黄色い線の方向と領域の方向 Di は一致する。

big_map[26][nab[mapmap][seg]] は図4の白い線の方向(筆触の向きDi) に対応しており、big_map[27][nab[mapmap][seg]] は領域の扁平の度合いを表している。この値が大きいほど領域が

表2　筆触表現に使われるパラメータ

パラメータ	数値範囲	説明
big_map[0][nab2[mapmap]]	0. ～ 255.	筆触のL*値
big_map[1][nab2[mapmap]]	0. ～ 255.	筆触のa*値
big_map[2][nab2[mapmap]]	0. ～ 255.	筆触のb*値
big_map[3][nab2[mapmap]]	0. ～	筆触に含まれる画素数、つまり、筆触の面積
big_map[4][nab2[mapmap]]	0. ～	画像の横幅　筆触中心座標のx値(pixel数)
big_map[5][nab2[mapmap]]	0. ～	画像の高さ　筆触中心座標のy値(pixel数)
big_map[26][nab2[mapmap]]	0. ～ 15.（16=方向なし)	筆触の方向
big_map[27][nab2[mapmap]]	0. ～ 1.	筆触の細長さ

細長くなっていく。big_map[3][nab[mapmap][seg]]は領域の面積に対応している。従って、3行目左辺のdirecf2、4行目のk4、5行目のk6をパラメータしてもつ6行目のdeltaXと7行目のdeltaYを自らの好みに合わせて変えることで、さまざまな筆触表現を生み出すことができるようになるのである。

5.2 新たな筆触表現の開発

前節で基本的な距離関数とそこで使われている変数を説明したが、ここでは、これらを色々と変えて新たな筆触を生み出す方法について述べる。アルゴリズミックアートとして多彩な表現を生み出すためには多くの人に新たな筆触を開発してもらい、このモードの番号が増えていくことが必要でる。ただし、ユーザーが勝手に増やしたのでは、多くの人に利用してもらうことはできないので、新たに作成した筆触はアルゴリズミックアートプロジェクトメンバーに連絡してもらい登録してもらうことにしている。ユーザーによって登録される筆触にはmodeの番号として30番以上の番号を割り当てることにしている。

プラグインのソース内には開発者の便宜のためにmode=30に筆触のひな形を入れてある。従ってこの部分を変化させることで、独自の筆触を開発し、登録の際には、ソースをそのままメールで送ってもらうことで登録ができるようにしてある。図6にはmode=30のソースを挙げてある。変数abstはGKReconstImageのプルダウンメニュー内で指定できる変数であり、規定値は95である。また、k4は1つの筆触がどの程度細長くなっているかを1から100の値で表現しているので、if (k4>abst) 以下の処理は、筆触がabstで示した細長さの値よりもより細長い場合に実行され、その結果その筆触はスケッチ線の様に細長く変化する。15行目のelse以下はスケッチ線のように細長くならなかった場合の筆触の形を記述している。ここで、17行目direcf2を

direcf2=0 (1)

にすると方向の情報を捨ててしまったので、円に近い筆触の画像が作られる。その結果を図7に示した。次に、direcf2はそのままにして、k4には

k4=k4/2. (2)

を用いて作制すると、方向性はあるが、ずんぐりとした筆触の表現になる。

筆触の表現をいろいろな条件によって変えることでさまざまな表現が生み出される。

先に示した例では、筆触の細長さのパラメータが使われたが、そのほか利用可能パラメータを表1に示す。

図6の4行および5行から分かるように、

```
if(mode==30)
        {
        k6=pow(big_map[3][nab2[mapmap]],0.5);
        k4=big_map[27][nab2[mapmap]];
        if (k4>abst)
         {
         k4=1;
         if(1./direcf2>70.)
            direcf2=0.19;
          else
            direcf2=0.19*(71.-1./direcf2);
         deltaX=deltaX*(direcf2*k4)/k6;
         deltaY=deltaY*(direcf2*k4)/k6;
          }
        else
         {
         direcf2=2.*pow(direcf2+0.0001,0.5)-1.;
         deltaX=deltaX*(1.18+direcf2*k4)/k6;
         deltaY=deltaY*(1.18+direcf2*k4)/k6;
          }
        }
```

図6 gk_reconst_image.cの部分 (mode = 30の筆触表現)

big_map[27][nab2[mapmap]] > abst

が条件であるが、これと同様に、表2に示した変数の大小を判断する基準とするための値を入力できるように、abst3、abst4が用意されている。今のところabst3のみがmode=14で使われているが、abst2、abst4は使われてない。これらは、表2に示した変数の閾値を入力するための変数として将来のために予約してある変数である。

5.3 筆触の彩色

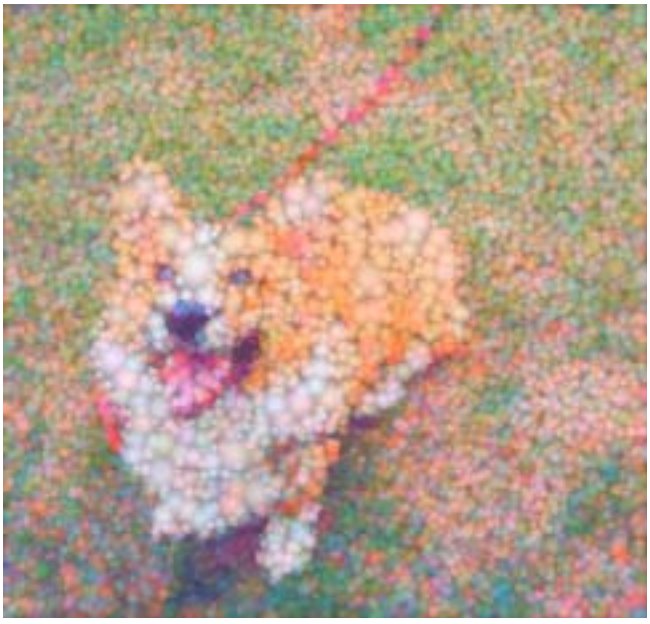

図7 direcf2=0として作成した例

最後のプラグインGKSubstColorは彩色のプログラムであるが、これも同様に一部分を変更することで大きな変化が現れる。もとの色をそのまま使うのであれば、画素の番号dotのL*,a*,b*の値をimage[0][dot] ,image[1][dot] ,image[2][dot] の値にそのdotが属する領域segの平均的なL*,a*,b*の値、つまりbig_map[0][seg]、big_map[1][seg]、big_map[2][seg] をmapから拾ってきて代入するだけでよい。つまり、

```
image[0][dot] =big_map[0][ seg ];          (3)
image[1][dot] =big_map[1][ seg ];          (4)
image[2][dot] =big_map[2][ seg ];          (5)
```

dot=画素の番号、seg=領域の番号

とすれば良いが、これを諸条件にあわせて変化されることで色に関してはさらに多くのバリエーションを作り出すことができる。図8はプラグインGKSubstColorのソースの一部である。GKReconstImageと同様にアルゴリズムアートプロジェクトに参加して、自分らしい筆触の彩色を作り出す際に用いるひな形を示している。

まず、6行から11行までは筆触の細長さがabst/100より細い場合はL*の値を半分にするという処理を行っている。つまり、GKSubstColorでスケッチ線を作り出したが、ここではそのスケッチ線を暗い色で塗り込むという処理をしている。次に12行から17行までは、18行から23行までの処理で統合された筆触(表現上はべた塗りされている様な表現になる)の上にどの程度の筆触を乗せていくかを表している。

以上の記述で分かるように、筆触を1つずつ塗り分ける場合は筆触の平均Lab値big_map[0][*], big_map[1][*], big_map[2][*]を代入し、統合された筆触つまりべた塗りにする際には、merge_map[0][mapf[*]], merge_map[1][mapf[*]], merge_map[2][mapf[*]]を代入する。ここで、merge_map[*][*]は統合された領域の平均的な色を表しており、mapf[*] は注目している筆触がどの統合筆触に含まれているかが分かるように、両者の番号の対応表を表している。

統合筆触と個別の筆触がどのように表現されるのかを作品を作りながら見ていきたい。まず、図9-aはスケッチ線なしで塗り絵のようなべた塗りの表現である。図9-bもスケッチ線はなしだが、塗り絵のようなべた塗りの表現の上に色をばらつかせた筆触を重ねてディテールまで表現した。さらに、図9-cはその上にスケッチ線を重ねて動きを感じさせるように工夫をした表現である。黒のスケッチ線が画面全体を締めている。

このように、基本的にdeltaX、deltaY、(3) 式～ (5) 式とそれに関係した多少のパートを変更することでオリジナルの処理を作り出すことが可能になるのである。ここまでの表現は筆者が開発してきた表現なので、既に、プラグインの中にパラメータとして組み込まれている。さらに、いろいろな表現が可能なことを示すために、図10と図11には以上示してきた式を各種の条件で変化させることで作り出した作品の例を示した。

```
else if(mode>=17 )
    {
    for (dot=0;dot<pixels;dot++)
     {
      tmp4=(int)image[1][dot]+256*(int)image[2][dot] ;
      if(big_map[27][ tmp4 ]>(float)abst/100.)
       {
       image[0][dot] =(int)(big_map[0][ tmp4 ]*0.5);
       image[1][dot] =(int)big_map[1][ tmp4 ];
       image[2][dot] =(int)big_map[2][ tmp4 ];
       }
      else if(big_map[27][ tmp4 ]>(float)abst2/100.)
       {
       image[0][dot] =(int)big_map[0][ tmp4 ] ;
       image[1][dot] =(int)big_map[1][ tmp4 ] ;
       image[2][dot] =(int)big_map[2][ tmp4 ] ;
       }
      else if( filter==1 )
       {
       image[0][dot] =(int)merge_map[0][ mapf[tmp4] ] ;
       image[1][dot] =(int)merge_map[1][ mapf[tmp4] ] ;
       image[2][dot] =((int)merge_map[2][ mapf[tmp4] ] ;
       }
      else
       {
       image[0][dot] =(int)big_map[0][ tmp4 ] ;
       image[1][dot] =(int)big_map[1][ tmp4 ] ;
       image[2][dot] =(int)big_map[2][ tmp4 ] ;
       }
     }
    }
```

図8 gk_subst_image.cの部分 (mode = 17の筆触表現)

6. Synergistic Art Project

以上示したように、プラグインのほんの一部分のプログラムを書き変えるだけで、自分だけの処理を作制することが可能となる。アルゴリズミックなCGであることから、処理に用いたパラメータが記録可能であることもSICの特徴の一つである。できあがった作品の画像ファイルのコメントには、それぞれの処理で用いられたパラメータが保存されている。

もちろん新しく作った処理がどのパラメータのどの値に対応するかが登録されていないと、どのような処理かは分からないが、対応さえ分かっていれば、作品画像のコメントを読むことで、できあがった作品制作の過程を知ることができるわけである。

このようにして作品を一人で作るのではなく、一つのコミュニティーとして作品群を作っていくことを狙いとしてSynergistic Art Project[4)5)]が発足した。今までの芸術活動は制作については非

a 筆触の全てが統合されているべた塗りの表現

b 統合された筆触に個別の筆触を乗せてデテールも表現した作品

c さらに、スケッチ線も加えて表現した

図9 統合された筆触、個別の筆触そしてスケッチ線の組み合わせによって変化する表現の様子

comment=plug_in_gk_mean_algo-01-04-01-01-04-00-10-05-05-04*plug_in_gk_add_principal-02*plug_in_gk_imerge-00-080-09-096*plug_in_gk_analyse-00-10-07-030-030-09*plug_in_gk_analyse-02-10-07-030-030-09*plug_in_gk_reconst_image-01-4-80-2-0-100-000-075-000-1-02*plug_in_gk_subst_color-00-1-0-095-085-000-040-11*

図10　新宿西口に設置されたウエップカメラの画像をもとに制作した作品。立体派の表現を意識して作成

comment=plug_in_gk_mean_algo-00-04-01-01-04-00-01-05-05-04*plug_in_gk_add_principal-02*plug_in_gk_imerge-00-080-09-096*plug_in_gk_analyse-00-10-07-030-030-09*plug_in_gk_analyse-02-10-07-030-030-09*plug_in_gk_reconst_image-02-4-80-8-0-100-000-075-000-1-01*plug_in_gk_subst_color-13-8-1-100-000-004-002-10*

図11　Hudson river に設置されたウエップカメラの画像をもとに作成した作品。水面の筆触空の筆触の方向の対比を強調。

公開であることが多かったが、最近ではコミュニケーションアートとして、制作活動そのものも表現として公開することが増えてきた。このSynergistic Art Projectも一つのコミュニケーションアートとして、多くの人と意見交換を行いながら新たな表現をつくりだす可能性を秘めている。
また、SICはインターネット上で鑑賞されるような小さな画像からでも、ポスターサイズの大きな作品を作り出すことができるという特徴も備えている。そのため、Synergistic Art Projectのアクティビティーの一つとして、ウエッブカメラで見つけだした画像をもとに作品を作り出し、それをもとにコミュニケーションをおこなうという活動を行っている。図10はハドソンリバーに設置されているウエッブカメラの画像をもとに制作した川の反対側に見える摩天楼の風景であり、所有者から作品に対するコメントも頂いている。この作品を作ったウエップカメラはパン、ズームなどの制御が自由にできるため、好みの画面を作り出すことができた。図10と図11には、作品を作成するとその画像のコメント欄には作成に用いたプラグインのパラメータが記述されているので、そのコメントも作品の一部として表示した。ただし、実際のSynergistic Art Projectの作品では、画像の中にパラメータを書き込んでいる。

7. まとめ

はじめに、芸術と科学の融合が何故進まないかについて考察し、自己表現のために科学的な手法を用いることに慣れることが、これからのメディアアートには大切であり、それに有効と考えられるアルゴリズミックアートを作り出す道具としてSICを紹介した。SICがオープンソースであることの重要性についてのべると共に、SICは先人の工夫の上に作家自身のさらなる工夫をアルゴリズムそのものに盛り込みながら作画が可能であることを示した。この特徴を生かして、多くの制作者とコミュニケーションを取りながらCGとしての表現を広げることを目的にしたSynergistic Art Projectに関しても述べた。
SICを用いると、画像を作り上げる際に他の人のアルゴリズムを参考にするだけでなく、処理過程が記録されているため、他の人の作品の制作過程を自分の制作の参考にすることが可能である。
このようなオープンCGといえるような新たな表現活動のプラットホームとしてSICを多くの人に利用してもらうことで、芸術と科学両面に通じる人材が増えることを期待している。

参考文献とURL

1. たとえばassembly (http://www.assembly.org/) や Back to the Roots (http://www.back2roots.org/Demos/)など。
2. SICを用いてコンテスト等で入賞した作品の紹介はLinux magazine,株アスキー,2000年7月号,p.77.
3. 笠尾敦司、中島正之、“シナージスティックイメージクリエータ-描画システムを重視した絵画作成システム“、電子情報通信学会論文誌DII、vol.J-81、no.4、pp671-680、Apr.1998.
4. Synergistic Art Project やSICについての紹介は日刊アスキーLinux (http://www.linux24.com/linux/news/column/)
5. このコミュティーは工芸大のホームページを中心に進めていく予定 http://www.dsn.t-kougei.ac.jp/cp/SIC