芸術のためのプログラミング

木村　健一[*]・美馬　義亮[*]・柳　英克[*]

絵画を描くという芸術的な行為を補助するようなソフトウェアを試作した。このソフトウェアはあらかじめ人間が与えたルールに従っていくつもの絵画パターンを生成する機能をもつ。これらのルールを定める作業は一種のプログラミング作業とも考えられる。本稿では、そのソフトウェアがどのような考え方で設計され、実現されているかについてのべ、さらにそのソフトウェアが生成する画像の例を紹介する。

Programming for Artwork

Ken-ichi Kimura[*], Yoshiaki Mima[*], Hidekatsu Yanagi[*]

We are developing a software, whose purpose is to support human to create artworks. This software can create many pictures by following the user defined rules. The process for defining the rule on this rule can be defined as a programming. In this paper, we will describe the basic concept, its software design and implementation. We will also introduce the sample pictures that is generated by this system

はじめに

パブロ・ピカソやピエト・モンドリアンは、画家としての生涯をかけ、次から次へと新しい表現スタイルを求める活動を行った[1, 2, 3]。彼らは、芸術活動を単なる作品製作でなく、作品に対する視点を発見するためのプロセスとして捉え、絶えず新たなビジョンを生み出すことを望んでいた。また、現代美術史の流れ自体が、新たな視点の追求でもある[4]。

このような芸術家の作芸活動においては、「継続的に新しいものを作り、作品を吟味することで、さらに高い評価を得る作品を作る」という継続的な内省的プロセスが本質的になる。作品制作過程を観察しても、このプロセスは内省と呼ばれ、一般的な塑像や絵画の制作過程において、絶えず繰り返されているものである。しかし、芸術分野の初学者に対して、このような視点を持って作品作りを行うようにガイドを与えることは、表現のためには基礎表現技術を身に付けることが必要でまずそのために多くの時間を費してしまいがちで難しいことである。

我々は以上のような背景をもとに、絵画を対象にして制作過程における内省を活性化させるためのツール ***ThinkingSketch*** を開発している。

ThinkingSketch が提供する機能

このツールは **Java2** で実装されたオブジェクトをベースにした図形エディタである。直線、矩形、楕円、ベジェ曲線、自由曲線などを基本図形（プリミティブ）として持ち、いつの時点でも再着色や、グルーピング、コピー・ペースト、ファイルへの書き出し、読み込みなどが可能な図形編集のための基本機能が実現されている。これらの基本的な機能に加え、絵画的なパターンを自動生成する機能が、付加されているところがこのツールの本質的な特徴である。

ThinkingSketch においては、エディタと対話する人間（以後操作者と呼ぶ）が図形を直接的に描くというよりむしろ、***ThinkingSketch*** に多数の図形の組み合わせとなる絵を生成させる過程を繰り返す。操作者は自分の判断で望ましい図形を安定して出力できるようになるまで、オブジェクト生成のためのパラメータをコントロールするという使い方を想定する。

ThinkingSketch によって絵が生成される仕組みは、まず、絵を生成するための要素として、1)絵を構成する部品の形態、2)それらの部品の配置、3)部品の配色、の 3 つの「モノ」と「操作」を基本とする。用意された部品は時に乱数や規則に基づき、選択され、配置され、配色がなされるというものである。

1) 部品の形態: 表示すべき部品(図形)は複数登録しておくことが可能である。個々の部品としての図形の形態は最終的に生成される絵全体を特徴付ける重要な要素である。これらの

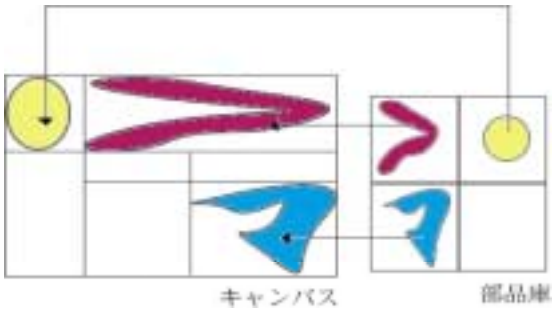
[*]公立はこだて未来大学 システム情報科学部, Future University – Hakodate, School of Systems Information Science

部品は、意図的にスクラッチの状態から作ることもできるが、抽象的な絵画を生成する場合には偶然の要素にゆだねても良い。

ここでは、操作者が画面の構成要素として使用するに十分な図柄を描けるだけの描画に関する十分な訓練を受けていない可能性をもつこと、さらに、システムに部品の生成を任せると生成される部品が一定の傾向をもつであろうという懸念ある。このため、今回のシステムでは、既存の画像をベースに操作者が図形を手作業でトレースする（なぞって、写し取る）ことにより部品生成の仕組みを提供する。

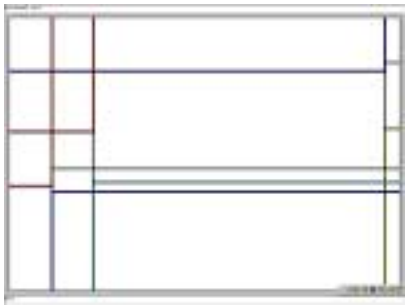
デジタルカメラで撮影した人物、風景やスキャナから読みこみ、JPEG、GIF などの画像ファイルとしてデータを与えると、それらの画像はエディタ上の壁紙として表示することが可能になる。操作者は、これらの壁紙の上で自由曲線などをもちいて図形を写し取り、新たな図形部品を定義することができる。部品は不定形な図形やグループ化された部品の集合の場合もある。これらの部品は、再利用にして「部品庫」に収納することが可能になる。

2)「部品」の再配置: 絵を表示するところはキャンバスと呼ばれる。部品としての図形は乱数によって決定されたキャンバス上の場所にその複製を配置することができる。これらの図形の配置はランダムに行われるが、操作者は、そのオブジェクトの配置方法について規則を与えることができる。このため、部品庫からキャンバスに完全にランダムに図形が配置されるのではない。すなわち、操作者が決めたり、選択したりした配置規則によって ***ThinkingSketch*** が生成する図柄には一定のパターンが現れる。



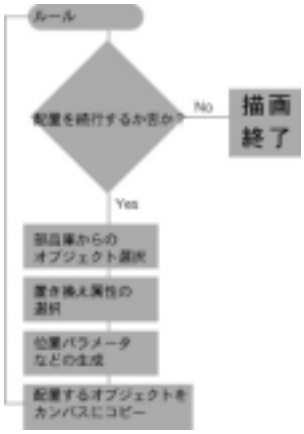
部品庫からキャンバスへのコピー

現在、実装されている部品の配置の規則には、既存のオブジェクトの上にだけ配置する、あるいは既存のオブジェクトのない場所にだけ配置する、グリッド上に配置する。配置の時に部品となる図形の大きさを変化させる、といったものがあるが、他にも配置をする際の条件として色彩や部品の大きさ、



あるいは構図における（上下、左右など）位置、向きの変更など図形の属性によって条件を指定することが可能であり、それらの機能は実装中である。

構図構成の戦略として代表的なものは、上記の図のように画面を乱数を用いて発生させた水平線、垂直線で分割したうえで、あるものは、画面の底部に置く、あるものは画面の左上部に置くといった単純なものである。これに加えて、配置を行う際に拡大縮小を伴うもの、図形のサイズを変更しないもの、上下あるいは左右に出現頻度を片寄せるものといったバリエーションを持たせた。たとえ、戦略は単純であっても、構成される画面の下方の隅に安定した図形が現れるということが、この構図戦略となる。この戦略を用いて生成された絵は、その絵を評価するものに安定的な印象を与えることになる。



3) 色彩の割り当て: 色彩の割り当ては、初学者にとっては形態生成と同様に難しい問題である。（観察によると、システムパレットのようなものをパレットとして与えた場合と洗練されたパレットを与えた場合では成果物の品質がことなることは初学者にも十分理解されるようである。）このため、部品の形態を決めたのと同様に既存の画像で使われている色調をそのまま利用可能な機能を実現した。

壁紙として読み込んだイメージの中から代表的な色彩を採り、これらの色彩で構成されたパレットを生成する。このようなパレットを用いて絵の部品を生成することにより、生成される絵画の色彩構成は比較的安定したものとなる。また、部品配置時に、既存の画像から採取

したパレットから、色彩を割り付けるということをもまた可能にしている。

以上のように、初期図形と配置規則、色彩割り当てが決まれば、これらの規則を繰り返し適用することにより、絵画（的なもの）の自動生成が起こる。その図柄は乱数によるばらつきはあるものの、操作者あるいは第三者からみて、生成された複数の絵画には一定のテイストが存在すると判断できることは興味深い。

対話的な利用と生成される図形

上に述べた機能は非常に基本的なものであるが、これらを組み合わせることにより、操作者が初学者レベルであっても比較的高いレベルの作品を容易に、しかも、大量に生成することが可能になる。このセクションで紹介されているのは、それらの例のいくつかである。

使用者はこのシステムと対話するうち、ごく自然に絵画のスタイルの存在に気づく。また、自分に絵を描く技術が無い場合でも、高速にオブジェクトを生成することのできるドローイングエンジンの力を借りて非常に多くの絵を、描かせることができる。そのため、その後、生成規則を変化させるという方法を用いて、自分の力でコントロールを行ううちに多くの一定のテイストをもった絵画群の生成が体験できる。

このように、「操作者が直感を用いて変更した規則がどのように生成する絵画を変化させるのか」ということをほぼ瞬間的に結果として得ることができるため、操作者とコンピュータはタイトなインタラクションのフィードバックループを構成しているといえる。操作者にとっては、このように高速なフィードバックが起こる体験は、創作活動として写真撮影とも異なる体験であり、鉛筆や絵筆による描画とも異なる体験となる。

同時に、このシステムでもう一つ特筆すべきことがあるとすると、それは一旦好ましいパターンを高確率で生み出すシステムが出来上がったとき、自分の与えた規則を一種のプログラムとして、客観的に分析することをも可能にするという点にある。

規則の一つ一つは、ある種のオブジェクトをどのような場所に配置するか、あるいはどのようなオブジェクトの近くには配置をするといった程度の小規模なものである。これらの規則を組み合わせることによって、操作者は希望するパターンに近い絵を、あるいは希望するパターンの絵をたくさん手に入れるわけである。

このとき望ましい形態を生成するための規則がどのように組み合わされているのかについてはシステム内に保存されており、これらは、操作者が明示的に与えたものと意識しなくても、エディタに蓄積がなされる、したがって、結果的に暗黙的に定まった規則から、たち現れるパターンの生成の原因は何なのかといったメカニズムについて内省をすることが可能になる。場合によっては、スポーツ選手が自分の姿をビデオで見て反省するときのように、自分ですら気づいていない「くせ」の存在やその正体を見出すことができるかもしれない。こういった、いわば「暗黙の知」や「感覚的」なものを言語化できるのは、このツールの大きな特徴のひとつである。

システムのより詳細な設計とその実装

ThinkingSketch の実装上の特徴について述べる。実装においては、自動生成された図形の一つ一つを直接操作を用いて自由に更新できるようにした。このような直接操作の重視は、この種のツールでは基本的な機能であるが、さらにその機能に加えて直接操作が可能な操作については、コマンドラインからの操作も可能となるようにした。

この結果、オブジェクトの生成、移動、選択、ファイルへの保存など、エディタ上の図形の操作のほとんどは「line 100 200 300 400」(点(100，200)に始まり点(300，400)におわる、直線の生成を意味する)のようなコマンドを与えることによって、コマンドラインから実行することが可能である。さらに一連のオブジェクトへの操作はテキストによりマクロ記述ができ、コマンドライン上から呼び出しが可能である。

本システムでは、その特徴のひとつとして、Java レベルでの拡張性を重視した設計を行っていることがあげられる。拡張は二つの方法によってなされると想定されている。拡張の

方法のひとつは、絵を描くための基本的なプリミティブを追加することであり、もうひとつは、プリミティブや部品図形をキャンバス上に表示するときのコマンド群に対する拡張である。

1）プリミティブの拡張においては、新規プリミティブはプリミティブの抽象クラスを継承するクラスを記述することにより Java のクラスレベルで比較的簡単に定義をできるように設計した。

2）コマンドの拡張に関しては、図形を表示する場である「キャンバス」の独立性を高め API を明確に定義し、それらを呼び出しを、Java 言語で記述することにより、自由にキャンバス上のオブジェクトに変更を加えることを許している。コマンドインタプリタを複数登録し、機能追加を容易にしたことなどが特徴として挙げられる。

現在は構図戦略や色彩割り当てといった、戦略記述の拡張は Java レベルでのクラスの拡張を行うことによってのみ可能である。しかし、前出のマクロ記述によってこれらの戦略を実現するための戦略プログラムを組み合わせ、あわせてプラグイン的な戦略プログラムライブラリの拡充により、操作者が図形生成をコントロールするための情報の多くの部分はカバーできるようになると思われる。

芸術のためのプログラミング環境

ThinkingSketch によるプログラミングは対話的な利用を前提としたものである。プログラミングという行為は表現という行為の中でできるだけ、意識されないことが望ましい。しかも、後にリフレクションが行えることも、必要である。現在のプログラミング環境は以下のようなものである。

1.プログラムによってデータを作り出すこともできるが、基本的なプリミティブはトレースや直接操作によってより多くのコントロールポイントを持つデータが作成できる。これらのデータはテキストデータとして変更を加えることも可能なフォーマットとした。

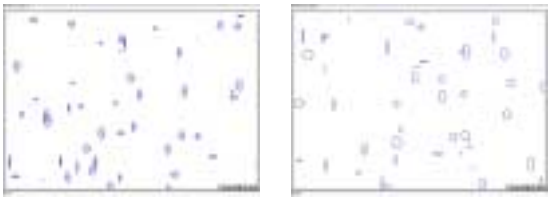
2.コマンドラインから入力するオブジェクトの基本操作ならびに部品庫からのマッピングなどのユーザの操作をまとめて記述するための実行機能がある。ファイルにまとめておけば、まとめて実行することが可能。

3.繰り返しについては実行前に繰り返し回数を宣言しておくという方法をとった。この方法をとることにより、for, repeat などの構文の利用を避けている。

4.実行においては、インタプリタは行単位での実行を行うが、エラーのある行を実行するときは、それらは単に無視するだけとなる。これらは、複数の任意のプログラムを単純につないで実行することを容易にしている。

評価と考察

ThinkingSketch は、まず、内省のツールとしての存在感は、期待以上のものとして実感できた。これらは、たとえば同じ配置戦略を用いても配置されるプリミティブが異なるだけで画面の雰囲気が大きく変化することの意外性からもわかる。



さらに、初期の段階で自由曲線で表現した図形が、簡単な割に抽象絵画として質の高いものを出力することがわかった。

このシステムを利用するうちに判明してきたのは構図や色彩の計画をエディタに任せることにより、図形生成の時間効率が非常に大きくなったことである。これらは、いままでアーチストが行ってきた作業に大きな変化をもたらすことが期待できる。すなわち、失敗を恐れずに大量に絵を生成し、そのなかから気に入った作品を利用することが可能になる。構図ばかりではなく、細かくてしかも機械的でない描きこみの作業などもコンピュータに任せることができる。たとえば、本のカバーや布地など「シリーズもの」のデザイン同一のテイストで効率よく生み出すことが可能になるのである。

関連した研究

乱数によって、芸術作品やデザインを行うという試みが 1960 年代にベル研のマイケル・ノルによっていくつかなされている**[5, 6]**。これらは特定の作家の作品傾向の分析とコンピュータによるアートの可能性を探ろうとする研究であったが、それから対話的なツールを作るまでには至っていない。

人間の芸術家のレベルと同等な絵を描くことのできるシステムの代表的なものはこのセクションで、写真により紹介した AARON[7]である。AARON は画家でもある Harold Cohen のテイストを実現できるように画家としての知識を備えたプログラムである。AARON について

は特別に設計された初期のころのロボット、あるいは後期に作成された XY プロッタに似たペインタによって、紙やキャンバス上に画家並みの作品を描くことができたことも非常に重要である。最近はこの AARON が PC 上で稼動し、ディスプレイ上に出力をするものが公開されている[8]。

AARON と ***ThinkingSketch*** の違いで一番大きな点は、AARON が作画におけるテイストを一つに絞りそのテイストをもつ作画を行おうとしているのに対し、***ThinkingSketch*** は操作者との対話により新しいテイストを次々に生み出す契機をあたえることを目的とする点である。

まとめと今後の予定

我々は、今後はこのツールをデザイン教育の中で実際に用いて得られる効果に関する検証を進めていく予定である。同時に ***ThinkingSketch*** は、考察の項で述べたように内省のツールであるとともに制作のための強力な生産性を発揮するツールにもなりうる。二つの視点を互いに強化していくことにより絵画生成ツールとして広く利用できるものとして完成度を上げてゆくことを考えている。

質疑応答

Q 伊知地(ラムダ数教研)：NHK のニュースのテーマ曲か何かで、ノイズをもとにした作品があったけれど、そういう作品のつくり方と ThinkingSketch との関係はどう考えるか？
A:芸術家の活動の中で、偶然に作用するものも多い。ノイズ的なもののなかからどういう意味を見出すかということが本質であったりする。したがって、ランダムなノイズをベースにした作品生成は ThinkingSketch の機構として中心的なものと考えてもよいと考える。

Q 伊知地:何でカタカナのテツ学、ゲイ術なのか。
A: いまの哲学、芸術では構えすぎであると感じる。これらの大衆化をはかりたい。

Q 伊知地: 暗黙知的なものの扱いはできなのだろうか
A: このシステムでは、ゲノムが表象として残るので自分の作風のようなものが、明示的なリフレクションの対象となりうる。

Q 福地(東工大):こういった類のシステムは、作品よりもシステムあるいはシステム作成者の評価となってしまうのではないか？
A:ゲノムにオリジナリティがあると思われるので、アウトプットの評価はゲノムの作成者に帰属するものとなるであろう。

謝辞

この研究におけるソフトウエアの開発に対しては、情報処理振興協会の未踏ソフトウエア創造事業として平成 12 年度、13 年度にわたり支援を受けている。また、プロジェクトマネージャの竹内郁雄氏には、研究の方向性に関してアドバイスをいただいた。また、Harold Cohen 氏については、我々が初期の研究の方向を決定する前の時点で AARON の開発過程や構造について様々な情報を提供いただき、大きな貢献をしていただいたことを感謝している。

参考文献

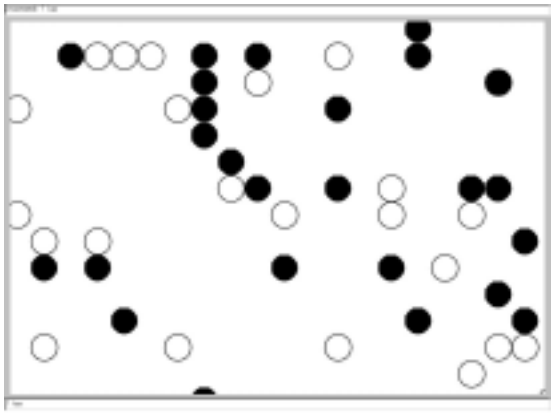
[1] まんが西洋美術史 3，高階秀爾 監修，1994 年，美術出版社
[2] PICASSO，PICASSO-MUSEUM VON BARCELONA，1996
[3] PIET MONDRIAN 1872-1944，Benedikt Taschen，1995
[4] 今日の芸術、岡本 太郎，光文社，1954
[5] Metamagical Themas，Hofstadter，Douglas R，Basic Books，1985，邦訳：「メタマジック・ゲーム」、竹内 郁雄、斎藤康己、片桐恭弘訳、白揚社，1990
[6] John Maeda MAEDA@MEDIA，前田ジョン，デジタローグ，2000
[7] Pamela McCorduck，AARON's CODE，W. H. Freeman and Company，New York，1991，邦訳：「コンピュータ画家アーロンの誕生」，下野隆夫訳，紀伊国屋書店，1998
[8] http://www.kurzweilcyberart.com/

付録：マクロによって生成される作品例

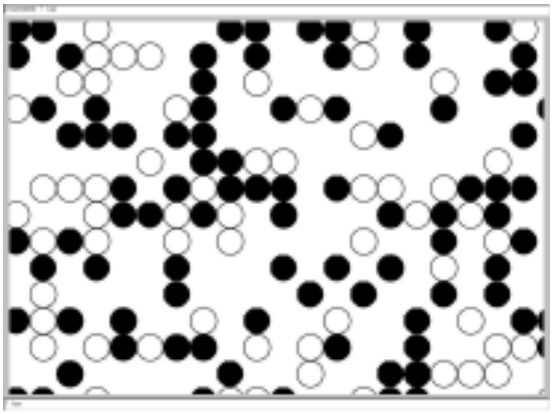
マクロ名　！ igo
マクロの記述内容は以下の通り。

```
color black  //オブジェの色を黒に

wmax 50  //オブジェを配置するエリアの
          // 幅最大 50pixel
wmin 50  //オブジェを配置するエリアの
          // 幅最小 50pixel
hmax 50  //オブジェを配置するエリアの
          // 高さ最大 50pixel
hmin 50  //オブジェを配置するエリアの
          // 高さ最小 50pixel

gridw 50 //オブジェを配置するグリットの
         // 幅 50pixel

gridh 50  //オブジェを配置するグリットの
          // 高さ 50pixel

repeat 25  //生成実行回数 25 回

fill fill  //オブジェは塗りつぶし
bubble     //オブジェは○
fill outline  //オブジェはアウトライン
bubble     //オブジェは○
```

！ igo マクロを実施にした作例



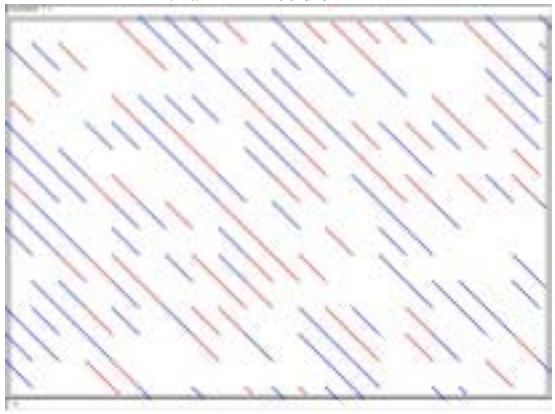
enter キーを押すことでマクロを繰り返す。
作業をしていると、囲碁の盤面を見ているような印象を受けるのでマクロ名として igo をあてた。
黒と白の○が enter キーを押すたびに微妙に配置を変えながら画面を構成していく。



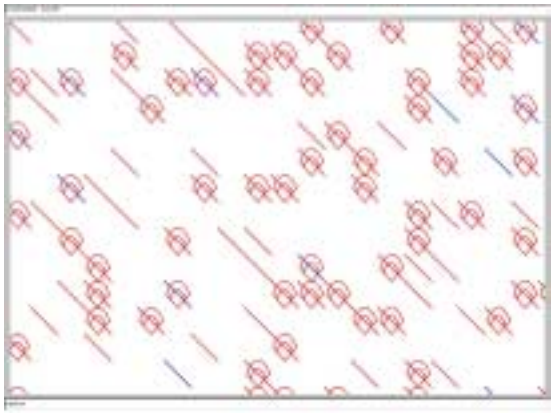
マクロ名　！ r
マクロの記述内容は以下の通り。

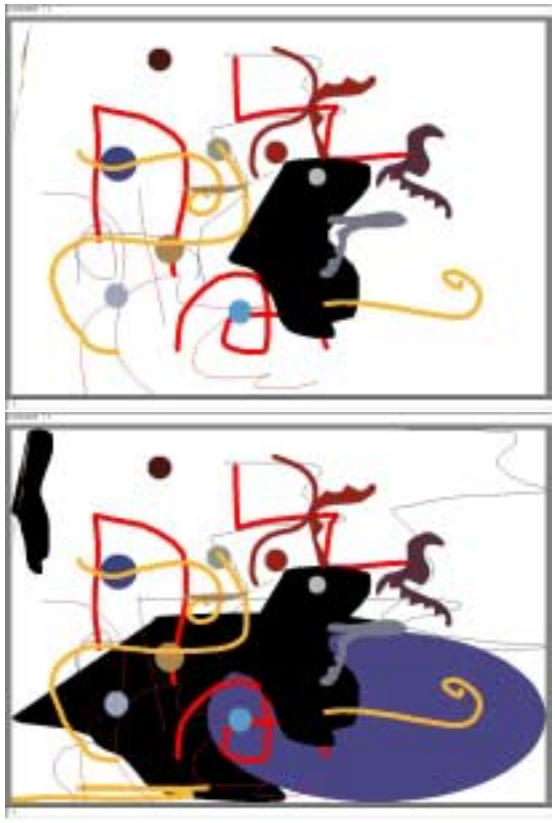
```
clear  //画面のオブジェをすべて消す

repeat 100  //生成実行回数 100 回
color blue  //そのオブジェは青
mode10  //描く場所にオブジェがあったら重
        //ねて描く
rain    //rain オブジェを描く
repeart 10  //生成実行回数 10 回
mode01  //描く場所にオブジェがあったら避
        //けて描く
color red //色は赤
repeat 300 //生成実行回数 300 回
rain  //rain オブジェを描く
```

！r　マクロを実施した作例



この作例に更に扇型のオブジェを生成する cooler コマンドを入力すると次のような作品が生成される。雨が降るようなオブジェクトを生成する rain コマンドと扇型オブジェを生成する cooler コマンドが画面上で交じり合うおもしろい表現が得られた。





マクロ名　！ t

マクロの記述は次の通り

import miro// miro という file を読み込む

showhide　　//画面上のオブジェを部品庫に
　　//しまう。

arrange　　//オブジェを部品庫から取り出
　　//し乱数で生成されたグリット
　　//に再配置する。

このマクロを使うにあたり、まず以下のようなオブジェクトをキャンバスに描いた。

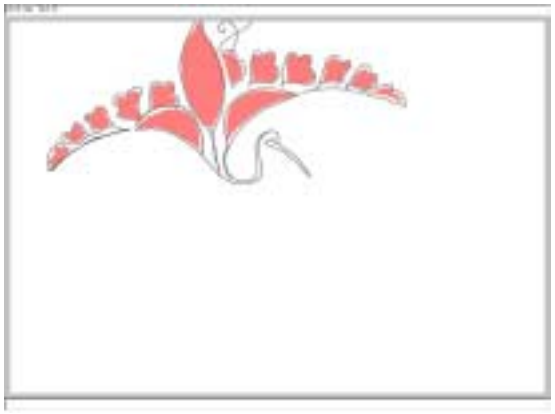
matisse と名づけられた file を import してキャンバスに読み込み。一旦 showhide コマンドで部品庫にしまい、cast コマンドで画面に再配置した作例を以下に示す。

オブジェクトを部品庫に保存 showhide を行い。その上で!t マクロを実施した例。

更に hito と名づけられた file を import して画面に賑わいを出した。

更に画面にあらわれたオブジェをすべて部品庫に一旦しまい（showhide)、cast コマンドで再配置を繰り返した作例。こうした試行を繰り返す中でマクロの記述を考えていく。

日本古来の文様を用いた場合の作画を次に示す。文様をスキャンし画像 file として保存。wallimage コマンドを使って画面に張り込み、オブジェを描くツールを使ってトレースする。



showhide コマンドでキャンバス上のオブジェが一旦部品庫にしまわれる。cast コマンドによって部品庫からオブジェが取り出され、画面に配置されていく。

文様本来がもっていた意味が、オブジェがバラバラにされる中で別の意味を持った抽象画に変わっていく。

原画には無かった、新しい味わいが生まれた。