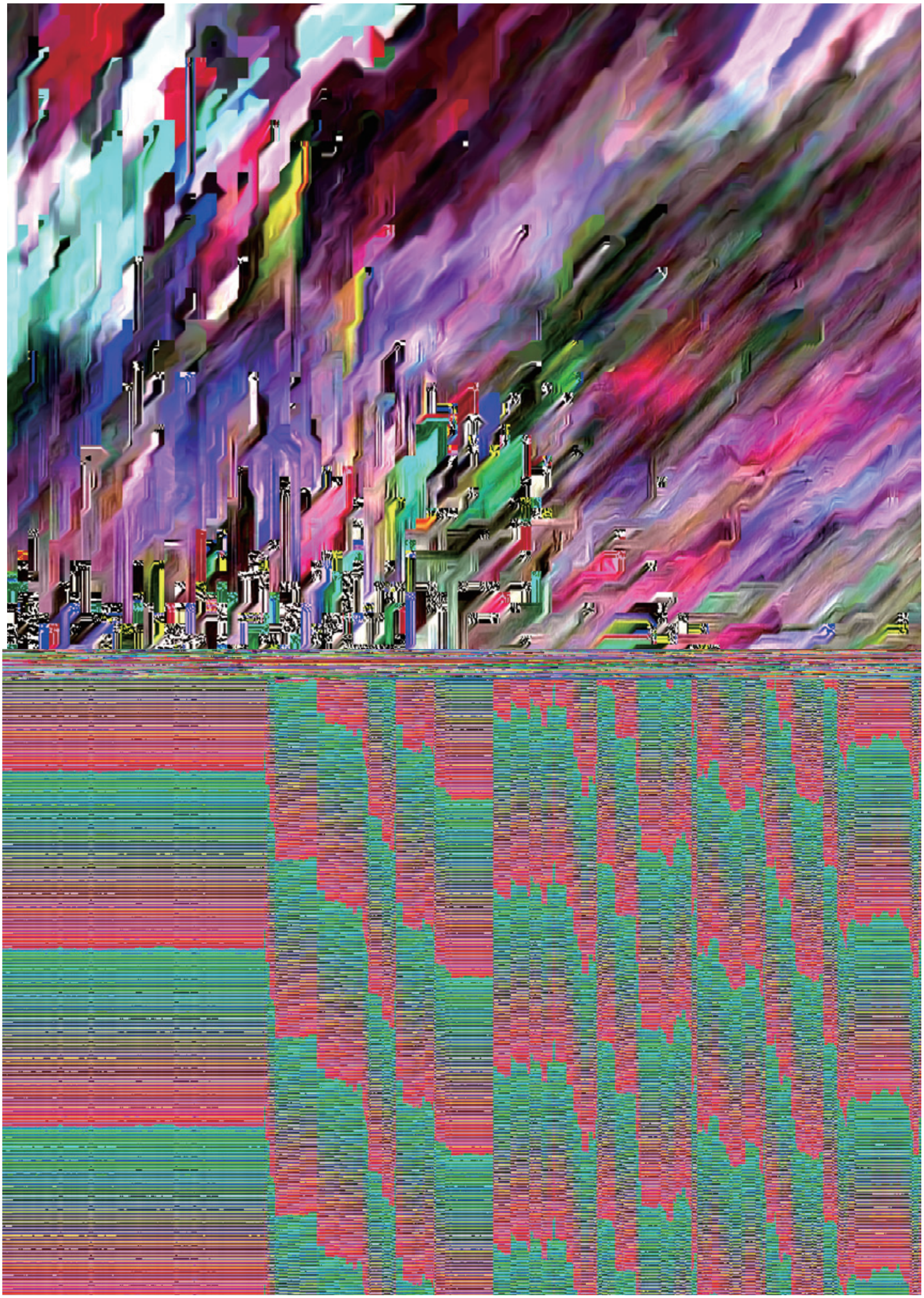

グリッチとは、単なるエラーや破壊状態の謂ではない。

グリッチとは、意図されておらず予測されていなかった状態が再生装置によって再生されることである。

たとえば Windows のブルースクリーンはグリッチではない、それは予測されているエラーだからだ。たとえばランダム生成によるホワイトノイズはグリッチではない、それは予測範囲内の結果だからである。たとえばある画像の各ピクセルを色値でソートし再構成した画像はグリッチではない、それは意図通りに生成されたただの画像にすぎないからだ。

ところで、ここではやむなく「再生」という単語を用いているが、再生とは play の意味であり、再生装置とは player の意味であるとあえて明示しておきたい。というのも、正確にはグリッチは「再」生ではないからだ。

ミュージシャン Oval は、盤面に落書きした CD を CD プレーヤーで再生し、楽曲として構成し始めた。彼らは音飛びの音像をグリッチと呼んだ。

コンピュータの普及途上で育った我々に一番なじみのあるグリッチは、ファミリーコンピューターのカセットを半刺しにしてゲームを起動したときのスプライトデータが異常な状態で画面に表示される現象かもしれない。テレビがデジタル化した現在に育つ子供らにとってはテレビ用のコーデックである MPEG2 が電波障害で破損したときに起こるグリッチが思い出として残るかもしれない。

写真のデジタルデータが欠損しブロックノイズとともに本来の写真には無かった色が現れた写真画像もまた、グリッチと呼ばれる。一方、破損したデジタルデータがビューアアプリケーションで読み込み不可能となりビューアが落ちたり、エラーダイアログがポップアップする状態はグリッチとは呼ばない。

グリッチと呼ばれる状態には、必ず再生装置が存在する。そしてそのとき、再生装置は意図されていなかった像を再生している。

再生不能ではないが、意図されている再生ではない状態。そこでは、オリジナルの像の再現という、データやメディアが再生装置に対して依頼した内容は、データやメディアが破損しているために正常に伝達されず、再生装置によって意図から逸脱した解釈をされ、予測されていない、想定からズレた像が、あらたに表出している。グリッチとは、オリジナルの像の中に存在していなかった像が再生装置によって生成されている状態である。そういった意味で、グリッチは「再」生ではない。

グリッチを規定しようとする人たちは、「真の」グリッチとは再現不可能だと言いたがる。つまり、その人たちはグリッチを体験したときの心象に重きをおいているということだ。同じ破損状態を完全に再現したとして、そこにあの驚きは見つかるか、という問いがあるのだ。彼らにとってグリッチとは、彼らの心の動きと分かちがたく結びついている。

一方そういった定義付けを行おうとする人たちは同時に、グリッチアートという言葉を使いもする。グリッチという客観的な現象ではなく、グリッチを見た人の心象に重きを置くからこそ、アートとなるのだろう。しかし、アートというからには再現性やなんらかの定着への指向があるはずだ。彼らは、自然発生したグリッチそのものと比較し、そのグリッチをタイミングよく録画した映像、そのグリッチと同等の現象を再現できる環境を再構成し何度も実行できるようにすること、またはそのグリッチの表層を模倣した複製としての CG を作成すること、などのグリッチ再現のバリエーションのひとつひとつに真か偽かのラベルをつける不毛な行為に手を染めていくだろう。

そうしたなかで、グリッチアートを標榜する人たちは、常に二律背反の中に立たなければならない。

さて、私はグリッチを展示しようとしている。

Turpentine / ucnv

ひとつめの映像では、MPEG4 規格の中に策定された動画コーデックで特に MPEG4 ビジュアルと呼ばれている圧縮技術を扱う。同等あるいは派生のコーデック実装によって、破損させたときの表情に差があるが、今回は ffmpeg が内部で使っているコーデック実装である FMP4 を用いた。

動画圧縮の基本は、フレーム間の重なりを省略するという方法である。キーフレームと呼ばれる画面内の全ピクセル情報、つまり一枚の画像データがまずあり、以降のフレームはひとつ前のフレームを参照し変化した箇所の情報しか持たない。何十フレームかそのような差分フレームが続いた後、またキーフレームが挿入される。その繰り返しで動画情報が構成される。現在普及しているほぼすべてのデジタル映像がそのような仕組みを持っている（ただし近年のコーデックでは前後のフレームからの差分をとる仕組みが採用されている）。キーフレームが定期的に挿入されるのは、なんらかの事故で動画が破損した場合にも復帰再生できるようにするためだ。そのような復旧の仕組みがあるため動画に対する闇雲なグリッチは一瞬で消失する。そのため、キーフレームという復旧の糸口を全体にわたって抜き取ってしまうことが、動画をグリッチする場合の基本的な処置となる。キーフレームを抜き取った動画、およびそのテクニックはデータモッシングと呼ばれる。

データモッシング動画は差分フレームのみで構成された動画である。差分フレームには色と動きの情報が含まれている。たとえば、あるピクセルが前のフレームから 2 ピクセル右に移動し若干青みがかる、といった情報が格納されている。再生装置はそれらの情報を淡々と処理しており、キーフレームが全く無くても、フレームの順番がバラバラでも、処理が可能なかぎり再生を続ける。当然のことながら、再生装置はそれが破損していることを知らないか、織り込み済みのエラーとして無視している。

今回用いるのは、ある差分フレームをコピーペーストし何度も繰り返すという動画グリッチの手法である。そのとき、本来の前後のフレームと切り離された結果そこに何が映っていたか判別できなくなり、意図されていなかった映像が生成される。差分が繰り返されることによって、あるピクセルは延々と 2 ピクセル右に移動し続け延々と青みがかることになるだろう。そこには、オリジナルの動画に含まれていなかったピクセルの移動と色が出現している。ここでは、シンプルなデータモッシングにあった、元の図像が崩れるといったことさえない。初見ではその映像がグリッチによるものかもわからないだろう。

もはやここに一般的な映像表現は無い。そこに提示されているのは、差分フレームの特性でしかない。デジタル映像を構成する要素のひとつがそこに表出しているだけだ。たとえるなら、油彩に使われるテレピン油がよく燃えるという事実を、それを燃やして見せることによって提示しているのと同じことだ。

ふたつめの映像で扱うのは、Huffyuv という可逆圧縮のコーデックである。可逆圧縮という特性上、このコーデックでは全フレームがキーフレームであり、それぞれのフレームがハフマンテーブルと YUV 色空間のアルゴリズムによって圧縮されている。

ところで、画像や映像のフォーマットは画面のサイズ情報を必ず持っているが、その面積に満たない少量のデータを再生装置にあたえたらどうなるだろうか？ JPEG 画像や GIF 画像では、基本的にデータが欧文と同様に左上から右下へ向かって順次処理されるため、上部は正常に表示される。情報が足りていない下の部分は、灰色の空白として表現されるようだ（ビューアアプリケーションによって色は違うかもしれない）。

Huffyuv では、若干様子が異なる。フレームのデータをそれぞれ、最初の少量のデータを残して削除してみる。そうしたものを再生すると、上部は JPEG や GIF 同様正常に表示される。ただ、データが無い部分は空白として表示されるのではなく、コーデックのアルゴリズムによって補完された、鮮烈なピクセルコントラストを持つ図像で満たされる。上部の少量の正常表示からの影響も若干見られるが、色や像は人間にとって全く意味を見出せない、空白という意味すら見出せないものになる。ここでもまた、元のデータに含まれていない、意図されなかった色や形があらたに生成されている。

再生装置に無があたえられたとき、圧縮アルゴリズムに従ったデコード処理が、本来の意味や作用から逸脱した有を発生させる。Turpentine という文字列から Internet という単語を抽出するような、無意味な連想ゲームを再生装置が行なっているかのようだ。そこに出現したピクセルは本当は存在していない。そこに表示されるのは、まるで再生装置の無意識である。