

Software we need:

RESOURCES <http://phillipstearns.wordpress.com/glitch-art-resources/>

Image Editor - Photoshop, GIMP, etc...

Hex Editor - HxD (PC), HexFiend (MAC), Bless (GNU/Linux)

Audacity

Introduction

Glitch art is the aestheticization of digital or analog errors, such as artifacts and other "bugs", by either corrupting digital code/data or by physically manipulating electronic devices (for example by circuit bending). (English Wikipedia Entry viewed 07-07-2013 - http://en.wikipedia.org/wiki/Glitch_art)

This simple definition captures the essence of a particular trend in digital visual art which incorporates digital (and analog) artifacts into the ranges of visual material for the production of art. There is already a lot here that we can continue to probe, but as we do, I hope to show that there is more depth to the discussion than surface features and their appropriation for artistic purposes.

First we'll have quick look at the definition of glitch and methods for understanding what is going on here. Next we'll deal with the notion of errors, artifacts and bugs. This will bring us to a brief discussion of entropy and the appropriation of entropic processes. After all this, we should have a wider understanding of the complexities of the topic of Glitch Art and how its ideas extend beyond surface aesthetics and reach into deeper critical dimensions.

What IS a Glitch?

Perhaps the best way to get the feeling of what Glitches are in the context of Glitch Art is to get into the process.

Databending – Text Editing? Let's Skip that and go Right into HexEditing.

1. Download and install a Hex Editor: HexFiend (Mac), Hxd (WIN), or Bless (GNU/Linux)
2. Do a Google Image Search and Download a JPG image
2. Open that JPG image into your Hex Editor
3. Scroll to the middle of the file
4. Select and delete a few lines of text
5. SAVE AS a new image
6. Open that image using your usual image previewer/viewer

What have we done? We have produced a glitch. But what exactly do we mean by this? Through one instance of opening up an image into a text based editor and removing some data, we have engaged in a glitch process. Through this process we can understand the Glitch to be some form of intervention. This intervention has to do with a reconfiguration of a material, in this case the material of the digital image, raw binary data. The result of such interventions/manipulations/transformations is a change in the image. Note that interventions can come from places other than intentional acts. Environmental agency in the form of noise or entropy can intervene with any structure in a way that produces an actual change.

Let's look at a few other things that we can do in a Hex Editor and their impacts on the JPG.

Manipulations:

1. Select and delete
2. Select cut and paste (repeatedly)
3. Select copy and paste (repeatedly)
4. Find and replace all instances of a single character
5. Find and replace all instances of a group of characters
6. Copy text or data from one file into another
7. Enter text or data manually (at random?)
8. Toggle between overwrite/insert modes

Not all interventions necessarily produce glitch, however. Some data can be altered without a noticeable change in the data. Is this a glitch? Perhaps not, if it's not noticed. Some alterations render the file unreadable. Is this a glitch? Perhaps, but this is one that is productive in other ways. The glitches concerning Glitch Art have to do with this place in between the business as usual and a complete breakdown. This is what I take to be one of many possible interpretations of what artist/theorist Rosa Menkman calls the tipping point of failure.

Important to note that the structure or anatomy of a file is important. The header is a part of the file which indicates to a program how to treat the data contained within. Although the file extension plays a role in what program is used to open the file, and even what files a program will allow you to open, the header contains some important data concerning the specifics of how it was encoded and how to decode it. Headers can be located in two places, at the beginning of the file and at the end. Best to avoid manipulating these for now.

When the alteration of the data produces noticeable changes in the image, we have something that we can name. This change in output is but one part of the Glitch, the artifact or product of intervention. The artifact results from interpretation of that intervention by algorithms involved in encoding and decoding data according to various criteria into a specified format. So our material here, is not simply the data, but these algorithms, they too can be manipulated to produce artifacts. This should come as no surprise because the only thing that distinguishes programs from data are the ways in which they are treated by the computational system.

In our JPG image example, the characteristics of the artifacts are due to the JPG compression scheme. Compression is the process of systematically reducing the data needed to represent a particular data set. Uncompressed images contain raw pixel color information; changing one bit may not result in much more than the slight shift in color of a single pixel. Compression can take this raw data and represent redundancies in a recursive manner that takes up less space. Here changing one byte may result in all pixels of one value shifting to another. Other compression methods may actually discard the original pixel information and represent attributes such as changes in chroma and luma in different regions of an image as set of values for parameters describing such changes. Here the original image is replaced by a parameteric representation.

JPG compression is an example of the latter form of compression. PNG, TIFF, BMP, GIF, PICT, PCX, etc. are file extensions which specify a set of standard formats by which an image is compressed for storage or transmission. Artists have figured out ways of developing their own image compression techniques as well as methods for altering pre-established compression algorithms or codecs. Kim Assendorf's Extra File project is an example of compression formats developed for creative purposes. Nick Briz's Glitch Codec Tutorial takes us through the process of altering compression codecs for video.

If you're running Mac OSX, you can download ExtraFile and experiment with data bending images compressed with his formats. For those of you who want a challenge, follow Nick Briz's Glitch Codec Tutorial. I haven't tried it myself, so you'll be embarking on an adventure of your own.

Let's take a moment to explore some different file formats.

1. Use GIMP or another program (Preview, Photo Shop, etc) to convert the image that you downloaded into different formats. Try uncompressed TIFF, BMP, GIF, PNG. Also experiment with the different export options. Take care to indicate in the file name which settings you used.
2. Databend those files using different techniques.
3. Generate at least 10 different glitched images. Remember, if you can't open the file, or don't notice any change after databending, go back to the original and try again.

NOTE: Some files require that the header remain in a very precise location. For GIF, PNG, BMP and TIFF files in particular, try first making changes in the overwrite mode, taking care not to alter the size of the file and thus the position of the header information at its end.

Defining Glitch – Relocating the Moment Through Recontextualization

If doesn't look like there's anything wrong, can that mean that everything is OK? If it looks wrong, does it necessarily mean that there's actually something wrong?

Glitches are mostly a result of miscommunication or mistranslation when transferring data from one environment to another. (From GLI.TC/H WIKI viewed 07-07-2013 - http://gli.tc/h/wiki/index.php/Glitch_History)

A glitch is a technology based, perceived moment. (From GLI.TC/H WIKI viewed 07-07-2013 - http://gli.tc/h/wiki/index.php/Glitch_Definitions)

Rosa Menkman describes a glitch “as a (actual and/or simulated) break from an expected or conventional flow of information or meaning within (digital) communication systems that results in a perceived accident or error. A glitch occurs on the occasion where there is an absence of (expected) functionality, whether understood in a technical or social sense. Therefore, a glitch, as I see it, is not always strictly a result of a technical malfunction. (Glitch Moment(um))

Common to these definitions is the notion that a glitch is something that is noticed, the we can identify, and give a name. The artifact is that product which precipitates from an intervention. The glitch itself though identified with the artifact is not only just the artifact, it is this call to attention of what is at hand. There is a dimension of error involved which draws our attention. What Rosa points out is that we notice a glitch and identify it with miscommunication, a break, error, or accident due to a specific context, placing the emphasis on the fact that these assessments are already conditioned in the act of perception. The moment of a glitch exists within an environment that is constructed; we participate in it with out necessarily being aware of the rules governing it and how our expectations have been conditioned to accept it as natural. Communication (whether or not it's intentional) is one of the larger contexts in which we experience glitches. Like Heidegger's hammer, we don't notice language or the communication environment until it fails us or breaks or becomes the subject of investigation (especially as a result of the former).

So let's return to material. If we identify glitches to arise from error or miscommunications between data environments, it's likely that we're dealing with some form of communication system. According to Claude Shannons Information Theory, communication can be modeled by a process in which a message that is encoded into a signal is transmitted from a sender to a receiver. If any message can be encoded and is essentially arbitrary or contingent upon context external to our model of communication, then the physical material of our system involves the encoder, transmitter, signal, signal carrying medium (may also include storage, which may involve additional encoding/decoding

stages), receiver, decoder. Though this model described technological communications systems, it can also be used as a tool for modeling the sensory/cognitive/affective processes embodied within each of us. Every part of the system is a “boundary condition” a place of coupling, overlapping, interpenetration, locations or sites for the environment to effect the structures of the system in question. In data bending our JPG we intervened on the structure by altering the stored data structure. Once the data is processed, it becomes a signal, which can again become susceptible to interventions.

Shannon's communication model is useful but limited in very specific ways. It was developed as a means for approaching the problem of optimizing data transmission across noisy channels. Every channel contains noise which places an upper bound on the data that may be transmitted. Tactics such as noise suppression, error correction, and data compression are based generally around the assumption that noise is something to avoid, that it is counter productive. Noise is something that effects the signal by way of the channel, but also acts at every point of mediation. What about the content or the message? Its omission leads to the omission of other possibilities, that of encoding the message in for one particular format and then decoding using another. Though it may be wrong in one sense to mis en/decode, it offers another range of potentials, and not only those useful to the arts.

Databending With Audacity

1. Export your Downloaded image, or an image of your choice as an **uncompressed** TIFF.
2. Open Audacity
3. File > Import > Raw Data
4. Select your TIFF image file and click Open.
 1. Set ECODING to A-LAW
 2. Set BYTE ORDER to No Endianness
 3. Set CHANNELS to 1 Channel (Mono)
 4. Set OFFSET to 0
 5. Set AMOUNT TO IMPORT to 100%
 6. Set SAMPLE RATE to 44100 Hz
5. Click Import
6. Test Export
 1. File > Export (ctrl + shift + e)
 2. Select folder and enter file name. Simply add “_test” to the original file name. BE SURE TO ADD THE PROPER FILE EXTENSION.
 3. Set SAVE AS TYPE to “Other Uncompressed Formats”
 4. Click Options
 5. Set HEADER to “RAW (Header-less)”
 6. Set ENCODING to “A-Law”
 7. Click OK
 8. Click Save
7. Open the file you just exported <filename>_test.TIFF

What you should get is the image without any changes. This process of encoding from one medium to the other and back again enables us to see whether the process is destructive or preserves the original data. Though we may not be concerned with preserving the original data in all cases, here it's important so that we can see our influence.

Go ahead and press play and listen to the file. What we're hearing is the raw data converted into a number, which is then converted into a voltage used to drive the speaker. Something that you could try on your own is to import the same file using different encoding methods and different settings from what we just used. For now, we're going to have a look at the different manipulations we have at our disposal now that we're treating raw data as “sound”.

Note: Here we have to be careful of the header just as with the Hex Editing. This means that we cannot make manipulations that alter the header information or change the location of the header, i.e. by changing the length of the file, or by cutting and pasting the header elsewhere.

Manipulations:

1. Select, cut and paste
2. Select, apply an effect

Try different effects. When exporting your files, try to name them in such a way that you can keep track of the effects you applied.

Mixing images

It's possible to mix image (and video as we'll see later) data from multiple files as audio.

1. Using any old image editor, crop and resize two different images so that they have the same dimensions.
2. Export these as uncompressed TIFFs
3. Import the first image using prescribed method and settings.
4. Click in the empty space below the newly imported file.
5. Import the second image using prescribed method and settings.
6. Zoom into the beginning and end of the data. Note where the data looks similar. This data should not be altered as it is part of the header. If we mix two headers together, the result is an altered header, and broken file. To avoid this, select that bit of data at the beginning of either one of the files (but not both!). Then click Generate > Silence > OK.
7. Repeat step 6 for the end of the file.
8. Export by clicking File > Export, select the folder, choose a file name, add the appropriate extension (TIFF). Use the Export setting described previously.

This process can be performed with more than one image. It can also be used to mix non-image data into an image. Not only is it possible to treat raw image data as sound, but it's possible to visualize raw data by cramming it into an image. Allow me to demonstrate.

Visualizing Raw Data

1. Import a TIFF into Audacity using prescribed methods and settings.
2. Import any file into Audacity using prescribed methods and settings.
3. Trim the raw data from the second file so that it fits inside the image data.
4. Select the image data that overlaps with the raw file data, then click Generate > Silence > OK
5. Export using prescribed methods and settings.

When we open this image, what we should see now is the raw file data converted into an image. If you want to change the way that data is rendered, you must start by converting your TIFF using a different uncompressed method. It may for instance be possible to export your TIFF in a different bit depth color space or with an alpha channel, or using planar RGB instead of interleaved. This seems like a lot of work, and there is more than one way to skin this cat.

Photoshop and GIMP, as well as other image editors may allow you to import raw image data. That is, you can actually convert any form of raw binary data into an image. The settings are a bit limited to those that would normally be associated with common image formats, but this method is effective.

With the programming expertise of Paul Kerchen, I helped develop a Windows based tool to render

raw binary data in a number of ways that are not normally associated with image encoding formats. This tool allows one to choose the rendering method based on the resulting image given in a preview window.

Out of necessity of not having access to a Windows based machine, I worked with Jeroen Holthuis to write a program in Processing which will render raw data according to desired bit depth on a per channel basis, to set the width, but not much more than that.

One final method that is a bit clumsy, but very hands on is to create a TIFF file, filled with zeros, or black, and use a Hex Editor to paste raw data from one file into the TIFF. So long as you don't overwrite the header of the TIFF, you'll be able to render the data in the format of the TIFF.

Allow me to demonstrate:

1. Using Gimp, create a 1000x1000 px image with a black background.
2. Export as an uncompressed TIFF
3. Open in your Hex Editor
4. Note where the zeros begin and end. We don't want to damage the non-zero data so we must select a chunk of raw data that is smaller than the TIFF.
5. Open another file of a different type in the Hex Editor
6. Select a portion that is smaller than the TIFF and copy
7. Switch to the TIFF window in your Hex Editor and paste in overwrite mode.
8. Save the File AS something else.

Editing Sound as Image Data

One thing that you should be acutely aware of is that so long as we play by the rules of the file formats, we can bend their applications to do things they were not designed to do. It should come as no surprise then that what we've been doing with images can be reversed and applied to sound. If we open a sound file as raw image data, it's possible to manipulate the sound data using filters and techniques typically reserved for images.

1. Take an MP3 file and open it as raw data in GIMP
 1. File > Open
 2. Where it says "all Images" click and select "all files"
 3. Where it says "select file type" click and select "Raw Image Data"
 4. Now choose your .mp3 file and click open
2. Set "Image Type" to RGB
3. Set "Width" to whatever you want
4. Set "Length" to 99999999999999
5. Set "Offset" to 0 and Ignore the remaining palette settings
6. Click open

Here you can manipulate the data. Just stay away from the stuff at the very top. MP3 files don't have any important data at the end and this method of converting data into images truncates the last incomplete row of pixels so this may not work for all file types.

Try selecting, cutting, pasting, drawing in the file, smearing, blurring, inverting, all sorts of crazy things.

When you're done click File > Export choose a location and file name and change the extension to TIFF. Select uncompressed TIFF.

Now using your hex editor, open both the original MP3 and the new TIFF you just created. Make note

of the data at the beginning of the MP3. Now look at the TIFF. There is some extra stuff at the beginning that needs to be removed before you can play the TIFF as an MP3. Select and remove that little chunk of data, then Save As and MP3.

Open the GIMPED .mp3

So now that we've expanded our framework for appreciating the materiality of data and its malleability. In the next session, we'll look at how these processes can be applied to video media.