A Recipe for Analyzing a Computer-based Artwork

1. What do I have?

This stage of the investigation lays the groundwork for the rest of your inquiry. It helps you to narrow your focus down to the relevant files and begin gathering the information that will allow you to speak to artistic intent and style.

1.1 What files do I have?

How do I answer this question?

Use the tree bash command to make a file tree. If this isn't possible use screenshots of the folder structure. Ideally whatever you produce will be in a format that will allow you to easily mark things as relevant or not relevant for your notes.

1.2 What file types are present?

How do I answer this question?

Go through the file tree you assembled in the last step and make a separate list of all of the file-types that you see.

Ask yourself

- What is each of these file-types usually used for?
- Given what I know of this artwork does it make sense that this file type would be present?

This information will help us to determine which of the files are directly relevant to the artwork and need to be investigated further. This can also begin to give us insight into aspects of the piece that will need detailed explanation. If a file type seems atypical for the work it may help us discover a previously undocumented functionality.

1.3 Which of these files are important?

How do I answer this question?

The best way to answer this question is by process of elimination. Try to break the file tree down into three categories.

- Documentation
- Not Relevant
- Relevant

Identifying Documentation Files

These are files that offer instructions on how to run the code, context or diagrams related to the code. Mark these files by coloring their entry on the file tree orange.

Look for .rtf, .txt, .pdf and .jpg, .png or other image files that have human readable information. If you're able to open a file, read it and it seems like it's giving instructions or perspective on how the piece functions, mark it as documentation. These may come in handy when you are attempting to understand the lower level functionality of a piece.

Identifying Not Relevant Files

“Not Relevant” files are files that offer no insight into the piece and can be ignored. Mark these files by coloring their entry on the file tree red.

Look through the file tree you generated in the last step. .DS_Store files, bagit files with "bag" or "bagit" in the name ie. bag-info, bagit.txt, transfer related files, and hidden files (files that start with a .) can be marked as 'not relevant'. Files that are similarly just a list of md5 hashes can be marked as not relevant.

Identifying Relevant Files

“Relevant” Just means that the files should be explained in the report. Even if they end up not being used in the piece. They need to be investigated thoroughly in order to determine their functionality.

Begin by returning to the information gathered at Question 1.2 if a file’s type means that it

contains executable code it is automatically important to the investigation. If a file type raises some questions or seems out of place it is also relevant.

If your piece contains HTML

Start with the index.html file. This is the file that will be first loaded by browsers. This file is definitely important. Think of this file as the front door to a home. You need to enter through the front door in order to enter any other rooms in the house.

So when looking at the index.html file make note of what files it references. If a .jpg file appears in the index file it's an important asset. Make sure to mark it in green on the file tree. Similarly any html file linked to from the index file is an important file. Not only should you mark it in green but you should also treat it as a new "room" in your theoretical house and investigate it from top to bottom just as you did the index file, marking assets mentioned in green along the way.

If your piece does not contain HTML

Identify everything that would be able to be run as a standalone piece of code as important. Playable media or picture files that are not clearly documentation are also going to need to be investigated.

1.4 What do we know?

How do I answer this question?

This is where you read through the code for the first time and start building out an understanding of what the technical context of your code is.

Go over the files that you have marked as relevant to the investigation and make notes on what you know for sure about each of them. If the code is human readable, i.e it’s an HTML file or JS file instead of an .exe actually read the code and see what you can learn from it. If you have no human readable code skip this question for now and revisit these principles in Question 4.

This section can also begin to give you insight into artist intent and style. For example if the piece was made in the year 2017 but uses very old HTML tags or custom HTML tags this speaks to a particular artist choice.

Reading the Comments

Start by reading any available comments in a file. Comments are lines of documentation left by the developer. They are not executed and often appear grayed out in text editors. These can tell you a great deal about a piece.

For example developers will sometimes leave headers at the top of a file indicating the file's purpose or who wrote it, sometimes even linking out to where the original code came from if it was developed somewhere else originally.

Knowing that the artist used a boilerplate piece of java to create the piece may not seem that important but it helps to give insight into their process. If external sources are mentioned in comments, research them thoroughly.

Reading Variable Names

Now read through the code with an eye on variable names. For less technical people this may seem intimidating but it’s quite simple.A variable is simply the name given to a value in code. Anything to the left of an equal symbol is likely a variable.

As you read through the file, ask yourself.

- Are the variable names informative or obfuscating?
- Do the variable names tell a story or are they perfunctory things like, x,y,z.?

This gives you insight into what type of programmer the artist was. If the code is extremely clear, well commented and neatly formatted it’s because they took the time to make it that way.

HTML Specific Advice

Look for age markers, if they’re incongruous with the time period the piece was created this is worth noting and exploring as a question of artist intent.

- Do tags have style markings in them? If so, this is HTML3 or earlier. In tab style declarations, which look like <body color=’white’> were done away with in HTML4 in favor of css.

- Is there a <style> tag? This indicates CSS is present which means this was make after CSS was invented in 1996
- What is the <DOCTYPE> declaration at the top of the file? This can explicitly date the file
- Are there any <meta> tags? These tags are used to indicate key words and help the site be indexed so they give insight into how the developer thought about the site.
- Are there any old or non standard tags? This requires a bit of knowledge about HTML or some googling but it can be incredibly helpful. Uses of tags like <blink> or <marquee> date the piece to a specific time period and the creation of unique tags like <my_personal_tag> would indicate a purposeful artistic choice.

Media File Specific Questions

- Is the format recent or legacy?
- Is it corrupted or are you able to view/play it?
- Can you run a media conch or similar metadata analysis report on it?
- Is the name informative, are things named in sequence?

2. What am I missing?

The detective work begins! This is where you begin to deeply dig into the files you're investigating and explore the larger landscape for other versions of the piece. After this question you should know if there exists an artist hosted version or "ground truth" of the piece. You should also be able to tell if you are missing any assets.

2.1 Are there any files or assets missing?

How do I answer this question?

Go through the code on a granular level, checking specifically for references to file paths and images. Is there a path mentioned that doesn't show up on your file tree? Is the piece linking out to something externally hosted? This counts as "missing" since you don't have control over such elements. If and when the external site goes down that part of the piece will be lost if you don't make an effort to download it and include it in your report.

2.2 Is there an artist version of the piece that you can access?

How do I answer this question?

This is your chance to do some archival research, check the way way back machine, use google, ask your peers, use whatever is at your disposal to find out if you can view the artist hosted version of the piece. If you can't find it then skip this whole section.

If there is an artist version of the piece

Go through the steps outlined previously and answer them again for the files you are able to pull from the artist site. After doing these things ask yourself the following questions.

What is the same/What is different?

Identify any files that seem to be identical and compare their hashes. This can be done by using the md5 command line function on macs and linux.

If nothing is different than you can be assured that you have all of the code that you need to move forward to question 3. However if you find something that doesn't match then you can dig into why. Did the artist change something after providing code to your institution? Did they give you a different version intentionally? This will require a bit of research but can be a very valuable background when discussing artist intent.

3. Does it work?

This question is where we triage the piece. We identify how if at all it is able to currently be run and begin to consider conservation concerns.

3.1 Can I run my code without any intervention?

How do I answer this question?

This is going to require a bit of technical know how. By “without intervention” I simply mean can you run the code in the way that such code is meant to be run. Whether you’re able to run the code or not record your initial observations of what you can see.

It’s valuable at this point to make a screen recording in addition to writing out your observations. If you make a recording be sure to record information on what machine and OS you're using, how you’re viewing the code and what the date is. In addition to what you can see, take note of what you can’t see. Are there elements of your “relevant” files that you would expect to see but can’t? For example if you’re looking at HTML with a javascript file attached do you see evidence that that file is being executed?

Even if you can’t run the code at all it’s useful to write down the conditions under which you attempted to view the piece. This can save you and future researchers valuable time when trying to figure out what will or won’t work.

3.2 Can I run my code with an intervention?

How do I answer this question?

If you were unable to run your code in the previous step, now is the time to dig in and see what your options are with emulators. Some formats have free and commonly used emulators associated with them, like Ruffle as an emulator for Flash. Other deprecated formats might require getting creative, find old computers, spin up virtualboxes, whatever it takes to be able to view *something*.

Once you’re able to see the piece run, document exactly what methods you used, how you decided upon that method and why. Make a screen recording as you did in the last step.

4. How does it work?

This is where you really dig into the technical aspects of the code. Use what you've learned from the previous steps and the questions you've had along the way to guide your technical breakdown of the piece.

It's very likely this part will unearth things that cause you to go back and re-evaluate some of your assumptions made in earlier questions. That's good! Re-engage with the questions as often as necessary until you feel you understand the piece at a granular level.

4.1 Is the code human readable?

How do I answer this question?

In order to do a granular analysis of the code you need to be able to read it. If you're dealing with something like Flash or a .class file you will need to decompile the code first. Be sure to note how you decompiled the code or otherwise gained insight into its functionality.

4.2 What does the piece do?

How do I answer this question?

You want to have a general summary of the functionality of the piece. For example "the html page loads a non-interactive jpg", or "the flash animation opens a second window when the user clicks the smiley face icon". For complicated pieces it's tempting to immediately dive into the code at a variable level, but it's helpful to start big and work your way down. It keeps you focused on the material outcome of the code and the artist's intent.

4.3 How does the piece accomplish its goals?

How do I answer this question?

Start by outlining how a user can interact with the piece. What can they click on, what

happens when they take certain actions? Use screenshots to capture each interaction and it's resulting outcome if desired.

Now go over each of these interactions and identify what part of the code makes this possible. Depending on the scope of the report and piece this section may include line by line explanation of code or a general summary of how each action is accomplished. Make supportive diagrams when necessary. For example if the piece uses a complicated case statement you should make a diagram of the resulting decision tree. If Case 1 happens what is the material effect, what variables are changed and what in turn changes for the viewer.

Return to the information gathered in Question 2, if you identified external resources how and when exactly are they being fetched. If the piece is written in HTML you can inspect the network tab and console in the browser to see what assets are loaded and how those are used. If the piece is sending or receiving any data you should be able to explain where it's getting information from, what it's doing to that information once it is received and whether the format and or content of that information is artistically relevant. Is it a custom format the artist created? Or is it something standard like JSON?

4.4 Is everything operating properly?

How do I answer this question?

This is a particularly difficult question because it requires that you catch something that might be invisible.

If you were able to run the piece without intervention it's tempting to think that everything is operating exactly as it was intended. However there could be subtle errors happening that you're not aware of.

Go back to step 2, if there are references to external links make sure that site is actually still online. Understand the potential failure of the emulator to properly display something. For example don't assume that because something appears to be non interactive that it is supposed to be non interactive. Check the code and be certain that everything described in code is happening on screen.

Refer to the background materials gathered in step one and see if there is anything

described or recorded in that material that isn't visible in the piece now.

5. What are it's piece's health risks?

Going back over the information you've gathered over the course of the report, what are the long term risks this piece faces?

5.1 Are any of the languages at risk?

How do I answer this question?

What vulnerabilities are associated with the languages used in the piece? Are any deprecated or in danger of being deprecated?

5.2 How would we manage external resources?

How do I answer this question?

Use the information gathered in Question 2. Outline what changes would need to be made to the piece in order to move to hosting and referencing these resources locally.

This is critical to the long term health of a piece because external assets present a vulnerability in two ways. First if that site ever goes down, which it inevitably will, then the piece will no longer function as intended. Second if the viewer isn't connected to the internet then the piece won't be functioning as intended. This is why capturing any externally referenced resources is so important.

5.3 What would be necessary to run this piece?

How do I answer this question?

Return to your notes from step three. If the piece was unable to be run without an intervention what changes would have to be made to make that possible?

Is the exact cause of some of the piece's difficulty still unclear? Make a recommendation for what type of expertise would be necessary to diagnose the problem.