# Pirate Programming Interfaces: hijacking the stream

Pierre Depaz

Post-doctoral researcher in Medienphilosophie

Hochschule für Gestaltung Karlsruhe

*Today, I would like to propose a contribution reflecting on the acts of piracy happening at the programmable interface level.*

*In doing so, I would offer some thoughts around a particular class of software I call rhetorical software, on the various modes of regulation present in the socio-technical artefacts that are APIs, and on the possibility for alternatives in the context of dependency.*

—

On October, 23, 2020, `github.com/ytdl-org/youtube-dl` disappeared from GitHub.

*In the fall of 2020, the popular code sharing website GitHub received a DMCA takedown regarding a specific project they were hosting. The Recording Industry Association of America was alleging that the software was enabling circumvention of copyrighted material. More specifically, that it allowed bypassing the Digital Rights Management system embedded in certain media assets owned by legal entitities and delivered by certain media platforms.*

—

`youtube-dl` enables theft (?)

*The problem was that `youtube-dl` is a software that allows the downloading of virtually any media on the web (mainly YouTube, along with 1,225 websites, as a of May 2025).*

*By facilitating potentially illegal behaviour,* `youtube-dl` *was itself found guilty of illegal behaviour, and the association protecting the owners of the stolen assest (here, the RIAA and the MPAA) demanded that* `youtube-dl` *be rendered inaccessible.*

—

On November, 20, 2020, GitHub reinstated `github.com/ytdl-org/youtube-dl` and donate $1,000,000 to a developer's fund protecting those targeted wrongfully by DMCA takedowns[1].

*But it turns out that it was not so illegal. The techniques put into place by youtube-dl were found to be based on a legitimate use, and the example of using copyrighted content as test downloads (such as Taylor Swift's Shake It Off which, obviously, is not owned by Swift, but by Universal Music Group), was found to be misleading.*

—

## The ethics of programming interfaces

—

The ambivalence of online piracy[2].

*This case illustrates quite well the ambivalent relationship between technology and (online) piracy.*

*This stems from a tension at the heart of the computer itself. On the one hand, ownership as we know it is based on most goods in our economy being rival, meaning that one cannot own and use a good while another one also owns and uses that same good. Non-rival goods nonetheless exist, such as jokes, light, air, and melodies If I hum a melody to you, you now remember it "have it and can use it", while I still "have it and can use it". For a long-time, this mnemonic requirement of using this particular class of non-rival goods depended on their fixation on physical media (wax cylinders, vinyl records, tapes and CDs). This meant that a non-rival good temporarily became a rival good, by virtue of its media fixation, and thus ownership in its classical sense was preserved.*

*On the other hand, the computer is, since its description as a Turing machine, fundamentally a copying machine. It reads and writes, endlessly and at an extremely low cost. The computer can only function effectively as a technical artefact if it can copy data, if it does not presume anything about this data (the syntactic/semantic disjunction established by Shannon).*

*The tension arose once the transcoding abilities of digital media enabled the storage of such goods in electrical circuits. Everything that is online is, by virtue of being digital, dematerialized, only represented as fleeting voltage changes. Songs and images could now be copied infinitely, and their owners were faced with a new kind of problem, in which rival goods became non-rival goods.*

—

An ambivalence is also always an **alternative**.

*I hypothesize here that it is this ambivalent nature that prompted the creation of the DMCA, the US legal framework for the protection of ownership of media assets. Because in an ambivalence, there is always an alternative. In this case, being ambivalent about the copying of media assets means that it could be considered as something good, or as something bad.*

—

Online piracy is an ethical act.

—

*Whenever we ascribe a value judgment to a behaviour, we enter the realm of ethics, in which acts are evaluated along the larger framework of what is a good behaviour, and what is a good life.*

*That is, ethical behaviour exist within a broader structure, defining what is good or bad.*

*Consequently, an ethical act that is at odds with existing structures (i.e. that is considered bad, or wrong), is also always one that suggests an alternative structure.*

*This is what the lawyer Jacques Vergès termed "rupture defense". You haven't done anything wrong because you do not recognize the structures which enable the bestowing of wrong or right.*

—

Software enacts (suggests) ethical behaviour[34].

*But what is particular about software, is that it acts, while other media mostly only represent.*

*For instance, the scrolling interface of Instagram enacts the good behaviour of spending time on the platform. While the paged interface of Google enacts the good behaviour of attempting to be featured on the first page (from a webpage SEO* *perspective) and of limiting one's attention to, and finding answers in, the top results (from a user perspective).*

—

APIs regulate (force) ethical behaviour[5].

*APIs add another layer to this. They embed rules in mechanical operations, distributing access to resources. It enacts inclusion and exclusion.*

*As a brief recall, APIs include different aspects:*

- *authentication (whether or not you have an ascribed identity in the face of the system provinding access)*
- *authorization (whether or not such identity allows you to access certain kinds of content)*
- *data representation (whether or not the data is accessible, made visible, in the the first place)*

—

From east code to west code[6].

*How laws (i.e. rules on good behaviour) are coded in programs.*

*Even the DMCA takedown is a pull request.*

—

Fair use, and the flexibility of the law[7].

*Interfaces operate politically insofar as they automatically solve contentious problems (who has access to what?).*

*And they do it in ways that are often more rigorous than their legal counterparts.*

*Lassègue and Garrapon, on their work on the digitalization of justice, note precisely this: while rules might seem to be strictly enforcing behaviour, they are nonetheless* *open to negotiation. The place of judges, jury, appeals and jurisprudence show that the written legal text is not really code to be executed as by an operating system, but rather code to be* ***referenced****.*

—

`youtube-dl` is ethical software.

*So we come back to youtube-dl, to how its use was considered bad behaviour by particular organizations believing in a particular structure and configuration and ownership, and how it got into this controversy.*

*I would now turn to examine more closely how can software propose reconfigurations of existing interfaces, whether at the programming level, or at the graphical level.*

*By looking at two examples,* `youtube-dl` *and* `pipewire`*, I would like to suggest two different kinds of alternatives, two different kinds of postures vis à vis a particular class of interfaces of streaming media delivery.*

—

# youtube-dl, from stream to data

—

Most web services today are split between **graphical** interface and **programming** interface.

*To understand how youtube-dl works, we need to understand how YouTube works.*

*What we see when we open a YouTube web is a graphical template, a kind of form separate from content. On page load, a JavaScript bundle then requests information from the YouTube servers, which is acessed via its various APIs, in to populate the page with actual content.*

*When I requested a video via the web interface, the graphical interface does 91 separate requests to 41 different APIs. There is therefore technically no single API for YouTube, just like there is technically no single "algorithm" (this might differ if we look at it culturally).*

—

```
https://rr3---sn-i5h7lnll.googlevideo.com/videoplayback?
expire=1747342058&ei=iv4laLWPFd3Oi9oPxK63AQ&ip=194.94.134.248&id=o-
AKoS6Z5KYXB0Gu0jBdhl0ItJofi9OlBhi1EGhr7gKPFc&source=youtube&requiress
i5h7lnll%2Csn-
4g5e6nsz&ms=au%2Conr&mv=m&mvi=3&pl=17&rms=au%2Cau&initcwndbps=3227500
gCIEn0F5RFVslPZV4XGRDKU3QwFH5z1shS27sLnHe4z7t3&lsparams=met%2Cmh%2Cmm
```

*One of these API calls is interesting. It gives a chunk of data:*

*Here again, there is some ambivalence:.*

*On the one hand, the request is particularly obfuscacted undergoing a process of minifying (sometimes called uglifying) in which humanly-meaningful tokens are replaced by machine-meaningful tokens to protect sensitive information.*

*On the other hand, it is also optimized for speed and performance, affecting the quality of content delivery and, in a certain sense, accessibility. For instance, in order to deliver content to our screens, Google came up with their own data format, UMP, which had to be reverse-engineered:* *https://github.com/gsuberland/UMP_Format/blob/main/UMP_Format.md*

—

The ambiguous status of the stream.

*The stream breaks down a file, broadcasts it in this chunked format, and then a client reassembles it on the other side.*

*This use of this particular strategy, based on _chunks, implies that the graphical interface does the important task of re-assembling the different streamed chunks in what seems to be a coherent whole. Whenever it is not coherent, when it glitches, or when we see this little spinner, loader, this is when the illusion of "continuous content" disappears.*

*So YouTube is an assemblage of two interfaces: the API provides access to the location of chunked data to the graphical interface.*

*This particular configuration enables a tighter restriction on content. Even though the computer is a copying machine, and even though the browser copies all elements that it receives onto the client's machine,* ***these two interfaces only render content usable through their mutual articulation****. the data stream provided by the GoogleVideos API is most easily re-assembled on the client side by using YouTubes's graphical interface.*

—

`youtube-dl` moves from one interface to the other, **impersonating** a web browser, **extracting** the stream URL and **stitching** it back together.

*So there are three parts involved in the process of* `youtube-dl`*, in what happens to be in* ***interfacing with an interface****. It first impersonates being a web browser, making use of particular header strings in the initial HTTP request to a video URL.*

*Specifically, it pretends to be a Chrome browser, under the assumption that Google would most likely trust itself.*

*Second, it extracts the stream URL, backtracing from the video ID, through a combination of parsing the requested HTML and querying the public API, until it reaches the specific endpoint providing the media-encoded chunks of data (usually M4A or OPUS containers).*

*From there, it stitches together all of these separate chunks intoone single readable media file by offloading the task to FFMPEG, a multimedia muxer/encoder/decoder (the same software used by YouTube itself to encode media when users upload videos).*

—

Piracy as **rendering** of the stream.

*In other words, what youtube-dl does, is that it renders the stream. It does so in different sense of the term.*

*Understood in software terms, rendering means that it stabilizes a fleeting, unfinished form, a multiplicity of separate assets held together by a particular software, just like one might render three dimensional models from CAD software, or a song from a digital audio workstation. By turning a stream into a file, youtube-dl renders audio and video, fixing it onto the user's hard-drive.*

—

The rendering of resources **from** from the stream.

*In a more traditional sense, rendering is also understood as giving (giving up, giving back), sharing roots with the contemporary french word rendre. This sense invokes other senses of ownership, possession, values. Rendering is therefore not just a technical operation, but also an economical one, intervening into which resource gets allocated to whom.*

*Incidentally, traditional forms of piracy could also be considered as a form of rendering, as economic assets get extracted from the stream of sea trade into* *different uses.*

*This phenomenon of extraction also illustrates the transformative power of an interface. There is a concept in intellectual property called the* ***analog hole****. When an intellectual property gets broadcast into the analog world (i.e. we hear it on the radio, or we see it on a screen), then all digital measures to ensure the maintenance break down and become useless. By becoming visible, or hearable, it leaves behind a particular set of digitally-encoded rules, and can, through digital capture, enter set a new set of rules, a new economical circuit.*

—

The stream implies **temporalities** and **connectivities**[8].

*Another significant reason for, and consequence of, using youtube-dl to interface with the youtube API, is a shift in temporalities and connectivities.*

*The stream follows a logic of "Just-in-time", a principal that exists in both software engineering and in logistics. In order to optimize the flow of, respectively, information and materials, a just in time approach ensures that only the appropriate amount of information or materials is used made available to the computer, or to the user. The idea here is to maintain maximum speed and throughput of delivery. As such, it creates an illusion of smoothness and availability, termed* ***the always on convenience*** *by INC. Crucially, this work is performed at all times whenever a piece of information or of material is accessed, access which depends on a large invisible infrastructure of protocols, standards and format.*

*The main dependency, and the one from which youtube-dl users claim to want their emancipation from, is the dependency of connectivity. Like all infrastructures, the one supporting streaming only become visible once they break down, to quote Susan Leigh Starr. And the illusions they support with it. So the smooth operation of youtube breaks down as it looses network connectivity, and the familiar spinner appears, asking us to be patient as the normal order of operation resumes.*

—

Active extraction for different value creation[9].

*Finally, what youtube-dl does is that, by extracting the resource of the media asset from a particular infrastructure of content delivery, it also changes the status of the media asset itself. By being rendered, it also re-becomes a source material, rather than an end product. It can be manipulated in more complex, more productive and less normative ways than when accessed through the graphical interface of the Youtube web or mobile platform.*

*This change of status also transforms the user from a passive recipient into a potential active producer, shifting the value of the media content from exchange to use, and making it a potential participant in, amongst others, a creative economy based on copy, paste, remix and rework.*

*In a sense, then youtube-dl is an interface to remove interfaces, a paradox to which I will come back in the conclusion.*

_

## NewPipe, from stream to attention

*now we switch to a different kind of pirate interface*

—

YouTube isn't just a technical system, it is a **political economy**.

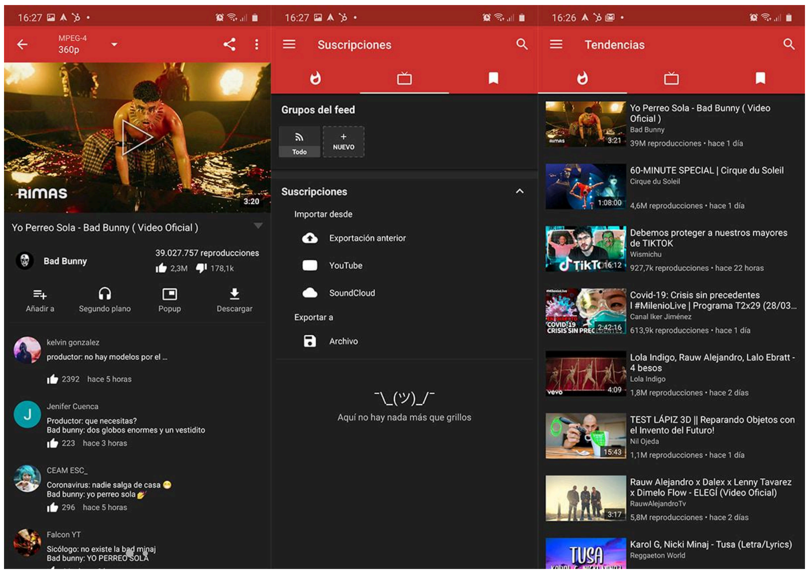
*Among the myriad of requests that YouTube does whenever one requests a video via their graphical interface, about half of them involve content which is related to tracking user behaviour (gads, gclick), delivering advertisement videos to be programmatically inserted in the vision of the video.*

*As such, the YouTube interface provides a different kind of access, specifically a different* ***direction*** *of access. It's not just for the client to have access to media content, but also for the client to have access to associated networks of data (profiles, comments, related videos) requesting that content. The graphical interface hosts the invisible JavaScript bundles which allow access to the client by Google, and its associated advertisement network.*

*Here, the resource management and extraction concerns the attention and the actualization of purchasing power, of the client, rather than the value creation of a copyrighted media content (although both are connected). The interface is here revealed as a two way mirror, and it can obfuscate more than technical assemblies.*

—

Pirating the interface itself.



NewPipe preserves the visual familiarity of YouTube

*NewPipe addresses this state of affairs by providing another interface, on top of the existing one. It is a browsing interface which accesses the youtube web interface, and does some modifications to it, among which:*

- *blocks trackers*
- *removes advertisements*
- *removes sponsored content*
- *enables background play*
- *…*

*So, it replaces YouTube with another version of YouTube, to the extent that it even mimicks the recognizable red color palette of the original service. But this new interface is one protecting clients, rather than exploiting clients*

—

Nothing but the stream(s).

*NewPipe requests the webpage identified by the URL, strips it of inconvenient, or undesirable elements, and serves it to the client, similarly to how an adblocker blocks particular domains from being requested an injected in the delivered content. In doing so, it maintains an interface of metadata, related videos and visual browsing.*

*Such kinds of interfaces are called wrappers, in that they completely surround a given system, without letting any of it appear to the user, but it also fundamentally depends on it.*

—

By replacing an interface, you replace **an ecosystem**.

*NewPipe might seem, at first like a "drop-in" replacement, a term referring to the kinds of technical solutions that can replace existing solutions wihtout any further modification or configuration.*

*It is an Android app, but it doesn't exist in the app ecosystem (not on Google Play, nor on App Store). The easiest way to install it is to do so through a completely different operating system (F-Droid, an alternative to Android). This is because the Android operating system is also owned by Google, and the company puts limits on the kinds of interfaces that are allowed within their ecosystem.*

*Different YouTube, different PlayStore, different operating system...*

—

Terms of Services and corporate legality.

*Indeed, in order to be listed on the Google Play Store one must:*

- *use proprietary Google APIs (such as the vibrate API, the media playback API, or the messaging API)*
- ***not** be free and open-source software.*

*Because, while it does not break any legal regulations (you are allowed to access and modify a webpage as you desire), it does break terms of services, corporate* *regulations. In this sense, using NewPipe is not so much about breaking the law of a country, but breaking the law of an organization, highlighting, just like youtube-dl, techno-normative entanglements.*

*Here again, piracy acts as a revealer of the systems that are being pirated. In this case, a deep intertwining of multi-lateral access, legal and technical systems, use and distribution. While youtube-dl collapses and renders all of these into a media file, collapsing them into one, so to speak, NewPipe only highlights even more these dependencies.*

—

NewPipe as an **anti-piracy** interface.

*Ultimately, we can consider NewPipe as an anti-piracy interface, meaning that it is a means of accessing a system without letting said system accessing the client, primarily for reasons of attention economy involving the gathering of personal data through the act of delivering a requested YouTube video.*

*Here, it shows that an interface is not just a means for a user to access a system, but that it could be recast as a two-way access, in which a computer system can also access a user, modelling it through browsing history, viewing habits, screen interactions.*

*All of these interactions are technical events that are transferred into capta points through an extractive process that has been rendered visible to most European users since the GDPR's requirement to display which cookies are injected in a browsing session.*

—

## Conclusion

–

Multiple critiques of a complex object, articulated around contestations of **technical dependency**.

*So, what does this dialogue of two alternatives tell us about the role, power and propositions of interfaces?*

*YoutubeDL and NewPipe offer two different kinds of critiques of YouTube as a socio-technical artefact essential to an extractive platform economy.*

*First, that the stream involves a (literal) network of technical dependencies through access to Internet, forced via Google's particular modality of client/server architecture, and of discursive dependencies, through relations with other, suggested video streams and the implied continuity between successive videos. Through accessing the interface directly,* ***youtube-dl*** *ultimately makes it irrelevant, by rendering the dynamic stream into static data.*

*Second, NewPipe underscores how the Youtube interface is not a unidirectional means of interaction, but that rather that its graphical component acts as a form of ideology, presenting an idealized reality (watching videos) and hiding material conditions of existence (being embedded in a advertisement network of attention economy).*

–

These acts of piracy ask the question of **the proper use of technology**[^ref-mumford].

[ref-mumford]: Mumford, Lewis. *Technics And Civilization*. Harcourt Brace Jovanovich, 1934.

*The historian of science Lewis Mumford, already acutely aware of technology's ambivalent nature, sought to address it through the identification of the conditions needed for the proper use of technology.*

*Given an API, meaning a regulated means of data access and interaction, what are the conditions for good of use of such a object?*

—

YouTube, `youtube-dl` and NewPipe all propose different answers.

*Faced with this question, the different technical configurations of the same function (distributing media assets to a client) seem to propose different answers.*

*YouTube's answer seems to be the obfuscation of the constellation of services supporting the parent company's core means of profit (advertising).*

*`youtube-dl` answer rather focuses on the extraction of the original data, hijacking the streaming infrastructure in order to render it into a previous, original state, manipulating interfaces to ultimately bypass them altogether, and enable a different circulation of this data. A good use is a "pure use", so to speak.*

*NewPipe's answer is complementary to youtube-dl's: it extracts the user from the political economy reified and distributed from the API.*

—

**Rhetorical software** is software whose syntax and semantics frame imaginaries and agencies[10][11].

*To frame the action of these softwares, I would like to suggest that these different positions operate in a rhetorical manner: that is, by choosing semantic and syntactic configurations that offer a particular framing on a given problem.*

*The syntax can be that of the visual interface, of what is shown, hidden, or referred to, through a combination of color, phrases and a register of possible (inter)actions. At this level, we can see very different approaches between, on the one hand Youtube and NewPipe, which favor a similar visual interface, at different levels of clutter, and on the other hand the strictly textual interface of youtube-dl, a command-line utility eschewing more "superficial" means of interaction with the interface.*

*As for semantics, we can identify the different entities foregrounded through these software: while Youtube offers an ecosystem, weighing on the relationality specific to digital objects (through recommendation, preferences, and user accounts),* `youtube-dl` *focuses on the primitive of the media object, the most previsible result of operating the software, and NewPipe mobilizes concepts of extraction, represented not so much in the operation of the software itself, but rather in its installation through sideloaders, means of bypassing official channels and countering the argument that YouTube can only be used with a Google account.*

—

Remains the question of **dependency**.

*Finally, these two rhetorical alternatives show a limit on their argumentation: they offer something different, but they still depend on YouTube, a coroporate platform that is ontologically first in this debate through software. Here, we see the limit of alternatives, which do appear to be provokative thought experiments but, in practice, cannot don the parasitic dependency that is inherent to both of these software, asking us to look for more sustainable alternatives to the platform economy somewhere else.*

—

## Appendix

1. Vollmer, Aby, *Standing Up For Developers* , https://github.blog/news-insights/policy-news-and-insights/standing-up-for-developers-youtube-dl-is-back/

2. Fisk, Nathan. *Understanding Online Piracy: The Truth about Illegal File Sharing* . Praeger, 2009.

3. Galloway, Alexander R. *Protocol: How Control Exists after Decentralization* . MIT Press, 2004.

4. Kerr, Ian R. *Digital Locks and the Automation of Virtue* . 2115655, Social Science Research Network, 1 Oct. 2010.

5. Snodgrass, Eric, and Winnie Soon. *API Practices and Paradigms: Exploring the Protocological Parameters of APIs as Key Facilitators of Sociotechnical Forms of Exchange.* First Monday, Feb. 2019. firstmonday.org, https://doi.org/10.5210/fm.v24i2.9553.

6. Lessig, Lawrence. *Code and Other Laws of Cyberspace* . Basic Books, Inc., 1999.

7. Lassègue, Jean et Garrapon, Antoine, *Justice Digitale* , Presses Universitaires de France, 2019.

8. Neves, Joshua and Steinberg, Marc, *In Convenience* in In/Convenience: Inhabiting the Logistical Surround, n. 54, Institute of Network Cultures, 2024.

9. Berry, David. *Copy, Rip, Burn: The Politics of Copyleft and Open Source* . University of Sussex, 2008.

10. Bogost, Ian. The Rhetoric of Video Games in *The Ecology of Games: Connecting Youth, Games and Learning* , edited by Katie Salen, The MIT Press, 2008.

11. Lavigne, Sam. *The New Inquiry* The New Inquiry, 3 Nov. 2017, https://thenewinquiry.com/dark-inquiry/.