ディスカッション

久保田晃弘：コードを記述し、実行し、保存する
KUBOTA Akihiro: Writing, Excuting, and Recording Code

登壇者：
永田 康祐（美術家/東京藝術大学大学院映像研究科）
市川 創太（建築家/ doubleNegatives Architecture Ltd.）
松川 昌平（建築家/慶應義塾大学環境情報学部准教授）

発言者：
藤幡 正樹、三輪 眞弘、赤松 正行

モデレータ：松井 茂

いま価値を判断すること

永田　市川さんと松川さんの研究や実践については、建築や都市のデザインにおいてコードをどのように使うか、つまりコーディングとその実行を通じて量と反復をどのように扱うかという問題に対するひとつの解として非常に興味深く拝聴しました。それは道具としてこうしたツールをどのように使いこなすかということだと思います。

いっぽうで、芸術作品としてコードを扱うことについて考えると、コードの道具的な側面というよりも事物的な側面、つまりコードをいかにユーティライズするかではなく、コードそれ自体の現れについて注目する必要があるだろうと思います。そして、そのときに問題となるのが、今回のシンポジウムのテーマにも挙がっている価値とか評価といった「このコードははたしてどういったものなのか」という判断の問題です。

僕は自分の発表のなかで、バルトをもじって「設計者の死」と言いましたが、書かれたコードがその文脈から切り離されたときに、実行する人がどのようにそれを理解し、価値判断を下せばいいのかというのはとても難しい。道具であれば、機能や性能の点からある程度判断することはできるのだと思いますが、そうでない場合は何をもって判断すればいいのかが曖昧になってしまう。これは扱っている領域が違うので、当然言葉遣いも異なるという話ではあるのですが、例えば松川さんの発表では、「判断」や「評価」といったものを定量的に扱えるものとして捉えられていると思います。しかしそれは、特殊なケースであると言わざるをえない。道具的に扱うことのできない対象について、こうした「実行的価値」をどのように検証するのかという点について、どのようなアイデアがあり得るのか？　ひとまずこの言葉をタイトルに掲げた久保田さんがどのように考えているのかお聞きしたいです。

久保田　そこそこがある種の主義主張、考え方だと思います。まず、自分で自分のことを言うと、基本的に僕は、懐疑的相対主義者で、何かをひとつに決めたくはないんです。だから、どうやってひとつに決めずに、常に相対主義的なところに居続けるかということを、いつも考えています。同じように、フィニッシュさせない、アンフィニッシュド（未完成）ということがすごく大事だと思うのです。ただ社会の中では、往々にして、フィニッシュさせなきゃいけない状況もやってくる。

永田　何かを保存する、つまり資料体として残すときには、その対象とある種のステークホルダーにならないといけない。これは多分アーカイブの問題だと思いますが、そのときに「いろんな見方があるよね」としてしまうと、「目に映るもの全てを保存しよう」というふうにしかな

らないけれど、それには限界がある。「これは残す」、「これは残さない」という判断が必要なシーンが絶対に発生する。そのときにどうしてもミニマルな価値判断が残らざるを得ないだろうと思います。そういったときに、相対主義的な判断をどこで切断するのか、というのは依然として難しい問題だと思います。

松川　そうですね、いまの価値判断をどう考えるのかという話はやっぱり面白くて、僕自身は、誰か特権的な専門家による価値判断ということをもう少し滑らかにしていきたいというふうには思っています。僕が先程説明したような仕組みの中でやりたいことは、価値判断そのものをもっと相対的に、僕から見た時にAという基準で見ると、Bという作品は50点だけれども、Cという作品を同じ基準で見たら60点で、同じ作品を見ても、永田さんから見れば50点というように、いろいろな人たちが価値判断をしていくと思うんですけど、その価値判断をずっと量子力学的な確率論的世界観というんですかね、そういう形で重ねあわせておいて、その確率分布の海みたいなものを、情報環境の中にそのまま保存しとくような感じがいいなと思っています。なので、どの作品を保存すべきか？　みたいな価値判断をするとき、その確率が高いものほど保存されやすいんだけれども、必ずしも保存される保証はない。確率分布だけをまずは仕組みとして提示することを考えます。

永田　補足させてください。先程言った、「残すものと残さないものを選ばないといけない」というのは、「この作品は残し、この作品は残さない」ということではなく、ある作品を残さなければならないとなったときに、この作品のどこまでを残すのかという問題です。その場合には、作品の良し悪しというよりも、その評価のしかた、つまり作品の何を重要とみなすのかということが問題になってくるのだと思います。

道具の範疇を外す

藤幡　乱入してすみません。この話は、要するに道具と作品をどう考えるかという話だと思うんです。そこを混乱しちゃダメ。

　目的がはっきりあれば、それは道具だと言っていいんだけど、しかし道具の部分が面白すぎて、そこからいろんなものが見えてきてしまう。だからその道具の部分も作品と呼んでもいいんじゃないかという話でしょう？　だから例えば、Photoshopが出てきて、バージョンが上がると、もう何作っても、そのバージョンのPhotoshopに見えてしまう、という時代があったわけです。メニューが決まっていて、メニューが入れ替わったりするから。そうするとPhotoshopを作っている人の方がエライというか、僕達はPhotoshopの手のひらの上で遊んでいるだけみたいなことになる。するとPhotoshopの方が作品じゃないか、とも言えるわけで、そういうメタレベル、メタな状況にデジタル・メディア以降なってしまっているわけでしょう。

久保田　そうですね。つまり僕は、コードを道具として扱わないことが可能なんじゃないかということを議論したいんですね。

藤幡　そうなったときにはじめてアートの議論になるわけですよ。

市川　藤幡さんが整理していただいたことから、松川さんの言われたことというのは、いろんな判断基準とか、価値があって、それを全部捉える、ビッグデータみたいな状態にしておいて、そこから何かを導いていくと。そうしたらまたそれを判断しなきゃいけないというか、なんらかのフィルターを通して、ビッグデータから抽出しないと何も出てこないってことですよね。すると、いつまで経っても終わりがないというか。専門家による判断というものを、よりオープンにしたい、広げたいというようなお話だったんですけど、私はどっちかというと、どうやって合意を取るのかだとか、そのためのプロセスとしてワークショップとか、あんまりいろんな価値観のアベレージを取っても、いつまで経っても中庸な判断しかできないんじゃないかなというふうに思っています。やはり専門家がもっと専門家になるというか、そういったいろんな見方もきちんと説明できるぐらいのポジションになっていく必要がある。みんなが良いと言ったからこれが「オッケーです」という考え方はあると思うんですけど、いやいやそんな方向に行ったら絶対にうまくいかないよ、という人もやっぱりいるんじゃないかなと思います。

久保田　先ほど相対主義と言ったのは、そこに《Super Eye》的な視点があるということで、それは価値判断における《Super Eye》といってもいい。だからコードというのはあらゆる状態を生み出す可能性のある、実行可能体みたいなものでいいと思っています。「実行する」ことは、音楽で言うと「マスタリングをしない」データという概念です。ただのデータだけで良くて、例えばサイン波のデータがあったら、そのサイン波がどう聞こえるのかは、再生したプレーヤーや環境による。それはサイン波というデータを作る人とは無関係というか、直接タッチしない状態にしておきたい。そこがポイントなんじゃないか、という気がするんですよね。

「『建築家なしの建築』の建築家」たり得るか

――　市川さんのお話は、永田さんの「設計者の死」という話、以前、松川さんでが『建築家なしの建築』（ルドフスキー、1964年）を目指している、という話とも重なるようにも思うのですが……。

市川　私も『建築家なしの建築』はすごく好きで、すごくいろんなインスピレーションを受けまして、実際モロッコ、トルコ、ペルーなど行って実物をみたりしています。一方で、アレクザンダーの『形の合成に関するノート』（1964年）というエッセイは、「建築家になってしまった以上、もうそんなことはできないんじゃないか?」というような話でもあるんですよ。つまり建築家という立場になって、こういろんなことを知った上であっても、まだ「建築家なしの建築」ができるのかっていう問いがやはりあって、そのようにボトムアップでできているということは、特権的に設計を決めないということなんですけど、決めないということまで専門家がコントロールするということでもあるわけですね。そのボトムアップのルールは、完全にその誰か、そのプログラミングした人とか、そもそも設計に沿って動くわけなんで、それはもっと詳細なコントロールなんじゃないかなというふうに思うんですよね。そういう違うプロセスでできたからといって、すごく広い意見が反映されてできたものになるのかなと。いくらビッグデータから抽出しても、その抽出のアルゴリズムというところにデザインが入ってくるので、ひとつのフィルターのデザインであって、それを設計した人はどこかにいて、ということは多分終わらないんだと思います。だからその専門性を間違えないようにする。優れたフィルターを考えるという方に注力するべきなんじゃないかなと思います。

松川　少し誤解があるようなんで、そこを補足した上で、意見を言いたいんですが、やっぱりビッグデータではないんですよね。先程の久保田さんのダイアグラムの中で、意味が作品の外側に出てましたよね。なので、作品そのものにはやっぱりまだ意味は属性がなくて、作品と観測者のあいだの相互作用に、作品を鑑賞する側の意味が立ち上がると思うのです。意味自体は、作品のなかに付与されていないから、その価値自体は、フラットなままデータでは残っていない。さっきやりたかったことっていうのは、みんなの意見を、最大公約数的に保存するということではなくて、またメタファーで申し訳ないですけど、盆栽みたいなことかなあと思ってます。半分は植物の自然の理によって育っていくんですけど、半分は専門家のトップダウン的な審美眼によって拘束するというか、その不断のやり取りの上に「建築家なしの建築」が立ち上がるのかなあというふうな感じを思っているんですけどね。

市川　そういった「オペレーションズ・リサーチ」は、究極に国をどうやってオペレーションするかとかというところまで行くと思うんです。いろんなところでフリーダムを与えてるんだけれども、総体としてうまくいくように、ちゃんとオペレーションされているというような見方もなくないですか?

久保田　そのときの建築家は、やはり近代的個人だとか、近代的な作家という概念と同じで、それはもう次に行かなければ、と思っています。近代的な個人だとか、作家というもの、あるいは建築家のように「××家」と「家」が付くものはいったい何様なのか、というところに対する問いだったりします。

市川　だからその「オペレーションズ・リサーチ」的なことが可能になってきちゃっているから、コンピューティングを操れる、現実的な計算量で実現できる人、あるいはソフトウェアがその位置にくるんですよね。さっきの盆栽の話もすごいよくわかったんですけど、一見ジェネレーティブなんだけどアンダーコントロールなん

ですよね。

松川　そうです。半分コントロールすることも含めて、仕組みそのものを建築家の専門性によって作ると。そういう意味で型を作る専門家になりたいということなんですけど、言い方変えると、「『建築家なしの建築』の建築家」たり得るか、という話ですね。そうありたいということなんです。

メタ化からベタ化へ

久保田　先ほど言ったマスタリングしない音楽家だとか、いま言ったような「アーキタイプ＝元型」を作る建築家を、いったいどういうものとして捉えればいいのか、ということなんです。いままではそうした最後のフィニッシュをする人が「××家」と言われていた。そこからマスターピースのような概念も生まれたし、あるいは建築家だったら最終図面に判子を押すような人、そこがなくなる。すると誰が困るのかな、と。困るというより、変わると言った方がいいのかもしれませんが。

市川　あらゆるグラフィックを作り出すPhotoshopを作る人ですよね、結局藤幡さんのお話も。

久保田　そうですね。そこをもう1度再定義して、顕在化させたいという気がします。

松川　ゆくゆくはメタ建築家も相対化していって、ポストモダン化してくと思うんです。だから型もきっとポストモダン化してくと思うんですが、いまの段階では形に建築家という作品性を付与してたと思うんですけど、その形のもう1個メタレベルの型に、作品性を付与できる段階なのかなあと。ゆくゆくはその型のレベルも相対化してポストモダン化していくと思うんですけど。きっとメタメタになっていく。

藤幡　おそらく果てしなくメタ化するんだと思うんです。

久保田　ポストという言葉をメタに言い換えてみる。つまり、ポストモダンをメタモダンにすると何が変わるのかな？

永田　「メタ」を言い出すと、永遠にメタ化の運動に絡め取られてしまうので、ポストとメタを併置的に使うのは危ないのではないかと思います。

すこし話が逸れてしまうかもしれませんが、例えば、ポストインターネットは、インターネットをメタ化したわけではなかったように思います。彼らの戦略の一部を単純化して言えば、「ハッキングからデフォルトへ」というものです。かつてのウェブには無限の可能性があり、さまざまな可能性が模索されていたけれど、それが商業化されたり、権力装置になっていくなかで、標準化という名のもとに高度にプロトコル化され、限られたことしかできなくなっていく。それをハッキングして自由を獲得しようというのが初期のネットアートの精神だったわけですが、彼らは、いちユーザーとしての地位に甘んじつつ、しかし批評的なユーザーであろうとする。それはハッキング的なメタ化に限界がきているということでもあります。

同様に、かつてCGには無限の可能性があったにもかかわらず、AdobeやAutodeskによるPhotoshopや3DS-Maxが出てきて、それらのソフトウェアによって設けられた枠組みの中でしか作れなくなっていく。それに対して抗っていかなければならないという状況は確かにかつてあった。しかしいまではもう画像を扱うんだったらPhotoshopがいちばん手っ取り早いツールになってしまっているし、それを使わなければ仕事にならない。そのような状況下でメタ化していくのはもはや不可能です。むしろ逆に、これは発表の際に提示した図式になりますが、ユーザーの側から抗っていくというようなアプローチもあると思います。それはつまりドゥルーズ、ガタリ以降のネグリ、ハート的な流れ、例えばギャロウェイのプロトコル批判とか、ダイヤー・ウィザフォードとデ・ピューターのカウンタープレイのような、ソフトウェアやゲームといった高度にアーキテクチャー化してしまったものをユーザーやプレーヤーの、設計者の予期しないような使用やプレイ、要するにバグ技みたいなものによって解体するようなこともできるんじゃないかと思います。つまりメタ化するんじゃなくて、僕はこのレトリックを好みませんが、「ベタ化」していくというような手法もあるんじゃないかと思います。

久保田　例えば、生産と消費、プロダクションとコンサンプションという言い方を止めるだけでも、工業化とい

うことから随分遠ざかることができますね。だから、ユーザーという言い方もやめた方がいいのかもしれません。エグゼキューターだとか、実行家というだけで何かが変わる。確かに、こういうアプローチもあるかと思います。

永田　例えば、Photoshopを濫用するみたいなことは、Adobeにまったく利さないわけです。彼らは、彼らのツールが完全に機能していると思われていて欲しいし、修復ツールなどで、変なグニャッとした画像が出力されてしまうような様を本当は見られたくない。こうしたソフトウェアを濫用することによって、そういうのを見せていくことができるし、それが表現として成立しうるということは、ユーザーというかエグゼキューターとしてできることだろうと思います。「実行的価値」は、コードを記述することに紐付いている必要はなくて、むしろコードなんて書かなくても、その実行における批評性に重要さがあるのだと思います。

コンピューターって何?

久保田　反射的に出てきた言葉なんですが、「使う」という言葉は、何となくヒエラルキーを感じさせてしまいます。でもソースコードのレベルでは、誰がクリエイターなのかは、そんなに明確じゃない。オープンソースのソフトを書く人も、それを改変する人も、そこにヒエラルキーはないわけです。どんどん変えていいわけですから。そういう何となく暗黙のうちにあるヒエラルキーみたいなものが、これも昔問題になった「私作る人、僕食べる人」というような関係と同じようにできてしまうとすれば、それは言い方だけで見方を変えることができるんじゃないかと。

　また、ハッキングがもたらした可能性というのは、設計者が想定しない使い方をすることで、そのヒエラルキーがなくなるということではあったと思います。

藤幡　商品化されてる状況の中だから、ヒエラルキーができるわけだね。70年代のUNIXコミュニティは完璧にイーブン。新しいアイデアで作った、新しいコマンドが、次のUNIX OSには載って出てくる。

久保田　そういう意味で言うと、想定外の使い方という言葉さえなかったわけですね。みんな想定内だったわけですね。イーブンということは。

藤幡　そうです。だから何でもいいんだけど、UNIXに「dc」という電卓のコマンドがあって、デスク・カリキュレーターかな。「dc」と数式を打つと答えが出てくるものがあると。次にもうちょっと複雑な計算ができるソフトを書いたやつがでてきて、そうすると次のOSのバージョンには新しい電卓が入っている。全然イーブンだったんです。その代わり、OS自体はかなり危なくて、よく落ちるだとか（笑）、新しいハードウェアが出てくると、みんなでそのUNIXを自分たちの力で入れるといった状態だったのね。でもその時でさえ、道具としてのコンピューターの「メタ」と、それを使って何かをする人がいたわけで、コンピューターの利用目的にはやっぱり違いはあった。

久保田　商品化と道具化を徹底的に押し進めたのがアップルでしたね。だからスティーブ・ジョブズがいつも言っていたのは、ユーザーがやりたいのは、コンピューターを使うことじゃなくて、絵を描きたいことでしょ、と。

藤幡　アップルのiPad Proのコマーシャルで、子供が「コンピューターで何してるの?」って聞かれて、「コンピューターって何」と答える。そういうことを徹底してる会社ですよね。彼はコンピュータを使っているとは思っていないというわけです。

コードというコンピューター言語

三輪　大変面白く聞きました。昨日、松隈さんが学生に建築のモデルを大変な苦労をして作らせるという、昔の手描きの図面を元に作らせるという事例を紹介していました。僕はとても感心したというか、確かに人が何かを理解するというのには、そういうプロセスは必ずあるし、必要だと思ったところがありました。今日の話は、同じく建築の話でしたが、さらにエキサイティングでした。僕が感じていることは次のようなことです。

　コンピューターを使って音を選ばせて音楽作品を作るとき、必ず問題になるのは、そうやって選ばせたコンピューターで電子的に演奏させれば理想の状態が聴けるじゃないかということです。なんでわざわざミスを犯す人間が演奏しなきゃいけないんだという、そういう考え

方があるわけですね。僕は、必ず人間がやらなきゃいけないものだと信じています。

そういう意味で、特に市川さんが、建築を手描きじゃなくて、表記の方法から変える、つまり言語を変えることで違う発想や違うアプローチを考えられるんだと。僕の勝手な想像では、実際の建築は、現実的な様々な制約や、施主の好みや文化や、ありとあらゆるものの辻褄を合わせて初めて成立するという意味で、非常に泥臭い側面を持っていると思うんですね。それは音楽で人間がやらなきゃいけないという、例えばバイオリン作品を書いたら、バイオリンが弾ける人がこの社会にいなきゃ成立しないわけだし、そういう文化があるからこそ成り立っているという、そういう意味ですごく、親近感を覚えた部分がありました。

一方で、久保田さんが仰っていた、自動生成した音列、またはパフォーマンスで、いつ終わるのかということに悩むという、ああこの社会に俺と同じことに本気で悩む人がやっぱりここにいるんだって思うわけですよ。だって終わる理由が無いんだから。音楽作品の場合、理由が無いので、僕は布教放送として、アルゴリズミック・コンポジションをweb上で延々と垂れ流しています。いままで職人芸や「何とか家」がやってきた特殊技能というものが、現実とのすりあわせというところで必要なんだろうなと、僕は思います。でも、建築や音楽を超えて、それがソースコードであるという意味で、やっぱり完全に別の言語でどんな分野でも書かれ得るようになった時代なんだなあということを思い、久保田さんがコードに注目してるっていうのは、人類が扱うユニバーサルな言語として、そこに焦点を当てた話題であり、今回展示している作品だったのだろうと僕なりに納得しました。

久保田　展示準備中に図書館で、三輪さんの《みんなが好きな給食のおまんじゅう》をずっと聴いていると、無意識のうちに覚えしまう。それで赤羽さんの展示を観に行くと、人が「実行」しているように見えてしょうがなくなってきた。つまり三輪さんが書いたコードを、人間が実行する。でも、いわゆるシリコンチップのCPUと違って、時々間違えるし、誤差もある、CPUが一生懸命コードを実行しているという意味で言うと、最初に言った、生物とコンピューターの関係が反転してきたことと同じことを、逆シミュレーション音楽からも感じました。つまり実行の意味を、コンピュータではなく人間に実行させる、と、ある種反転させたところがすごく面白いと思いました。

積極的な敗北主義を考える

―― 赤松正行さんにコメントいただきたいと思います。

赤松　IAMASの赤松です。今日は刺激的な話をありがとうございました。

最後にUNIXとアップルの話が出ましたね。シンポジウム初日にも所有権とアクセス権の話が出ていました。まさにコードは書けるけれども、実行できるかという問題に、2017年は瀬戸際の年だと感じています。というのも、例えば、いま話に出たアップルにしても、ジェイルブレイク(脱獄)が不可能だと、コミュニティが宣言してしまったり、単体ではインストールして起動できないMacがまさに今月発売されてしまった。もちろん、素朴な昔のデバイスを使い続ける、という選択肢もあるけれど、そうするとどんどん社会と乖離してしまう。つまりコードは書けるけれど、実行できない。あるいは、アーカイブはできるし保存もできるけれど、再現できない、という状況が2018年ぐらいから始まりかねないですね。そういうときに、永田さんのPhotoshopの話が、逆に「ブラックボックスで何が悪いの?」という、積極的な敗北主義のようにも聞こえたことに、僕はシンパシーを覚えました。そういう状況を考えたときに、UNIX的なカウンターカルチャーがいったいどうなるのか、ということを含めて、今後僕たちはその現実とどう向かいあっていけばいいのか、ということを、僕も特に結論はないんですけど考えました。

久保田　スティーブ・ジョブズがいうような、「世界を変える」という言説はもうとっくに終わったと思っています。イノベーションだとか、そういうことではなくて、こんな時代の中でいかに「世界に変えられないか」ということが大切なことなのだと思います。世界に変えられないために何をするべきか?　ということを考えています。「世界を変えよう!」って言った時点で、もう話が終わってるような気がするんです。つまりUNIXのようなカルチャーとは違う世界に行ってしまっていて、でも、コードを書けるようにしておけば、コマーシャリズムや

エンターテインメントのような、否応無く押し寄せてくるパワーに対して「いやいや、ちょっと待て」という余裕や余地を持てるような気がしています。「作品をつくる」ということも、もはや何かを表現するためのものではなく、世界に変えられないようにするための実践だと思っています。

再度「実行的価値」とは

―― 議論を経て、いま1度「実行的価値」という言葉に戻ってみたいと思います。

松川　難しいんですが、実環境で考えた時に、コードを記述するとか実行するとか保存するって、どういうことなんだろうなあってずっと考えてたんですよね。コードを記述するっていうのは多分あれですよね、万物理論みたいな数式に落とし込めた時に、その数式がエグゼキュートされると、今の宇宙ができあがるっていうような、そういう感じなのかなあと思って、なのでその場合なんか保存で何だろうっていうか。きっと保存って無いっていうか、ずっと、エラーも含めて実行し続けてるだけっていう感じなのかなあと思っていて、なので、僕達が行動を記述するということは、そのパラレルワールドみたいなものを想像しうる可能性を持っているという意味で、僕さっき生態系を記述するっていう話しましたけど、そこに面白さがあるなあと思っていて、なので実行し続ける、切断し続ける、まあ保存はないっていうかずっとパラレルに宇宙が多世界的に立ち上がってくっていうような状況が起こると、先程のようにもう誰ももう消去できないというか、自分の手を離れて、自分すらもそれを書き換えれないし、消去しようと思ってもできないっていうような状況が作れうる可能性を持っているという意味でコード面白いなと思いましたね。はい。

市川　すごい話がでっかくなってしまって、うーん困りましたね。久保田さんがすごくコードフェチなんだなというのはよくわかりました。自分は、わりと道具として考えているので、そういうフェチ＝愛情というのは無いですね。

　コーディングというよりは、自分はプログラミング行為の方がメインにあって、別にそういう特殊な言語を使わなくても、いろいろ日常的に目覚ましをセットするとか、ビデオを予約をするとか、そういうのもプログラミングだと思うんで、日常にもう溢れてるような行為だと思うんですね、プログラミングというのは。その詳細度や自由度のバランスで、もっとカスタマイズというか、もっといろいろできるっていうか。いろいろな専門道具、例えばカメラとか、専門性が高くてカスタマイズできるものほど、扱いが難しい反面、いろいろできるようになっている。そういうプログラミングの最たる結構なフルカスタマイズっていうところが、コーディングになっていくのかなあと思ってはいます。

　コードを実行するというところの、自分で書いたコードがうまく動いてるとか、そういう感触はもちろんあるし、それはそういうプログラミングができるようになっていく、コーディングを修得していくときに、やっぱりそういう感覚がどうしても必要だし、プログラミングをできる人はどんどんどんどん増えてった方が自分はいいと思うんですよね。あらゆることを、詳細にカスタマイズできるようになってくというか。だから、そのコードが動いてる喜びみたいなものが、それ自体が何か切り離されたものじゃなくて、どんどんいろんなことのレベルが上がっていくときに付随してくる、なにかちょっとこうパッパッと点くような火花のようなものかなあ思っています。

　全然話のステージが違うんですけれども、さっき松川さんがおっしゃったように、ある世界が全部こう、つながってるふうに考えると、確かに保存とか、切断とかっていうのが、世界が止まってしまうようなことになってしまうので、もう保存とかそういう表現ではないような気もします。コードというのはそこで閉じられた世界を設定することができて、コンピューターを使って、いろいろ実行する前に試せるとか、何回もトライ・アンド・エラーできるとか、そこにそれはすごく価値があるような気がします。保存するということは、先ほどの松川さんの言葉のように、パラレルに、いろんな状態を保存、キープできるとか、そういうところの価値ですね。そのとおりというか普通ですけどね（笑）。別に、そりゃそうだっていうことなんですけど、そこに独自の価値を言及するのは何か難しいという感じです。

永田　赤松さんからも所有権とアクセス権の話がありました。普段プログラミングをしているとXcodeが勝手にアップデートされていて苛立つことがあります。僕はよ

く、GoogleCloudAPIを使うのですが、GoogleCloudAPIも勝手にアップデートがかかって、ファンクション名が変わるといったことが起こる。要するに、自分の書いたコードは保存できても、参照してるAPIだったり、実行しているフレームワークが勝手にアップデートしてしまうから、実行可能な状態を維持するという意味での保存はかなり難しくなってるし、それをAPIの提供者がブラックボックスにしてしまったら最後、もうアップデートできなくなってしまう。

「糊としてのコード」という話をすこしだけしましたが、そういったコードの相対化が免れ得ない状況において、フルスクラッチというのはもうありえない。だからUNIXコミュニティのようなあり方には強く憧れるいっぽうで、その不可能さも強くかんじます。例えば僕がプログラミングを始めた学生のころは、Processingが普及して、これからopenFrameworksが来るぞ、というような状況でした。オープンソースのコミュニティが、コミュニティベースでアプリケーションを育て上げ、それを用いて作品を制作しよう、という機運が最も高かった時期だったと記憶しています。しかし結局のところ、そういったコミュニティが大きくなりすぎて、そこからアウトプットされる表現はほとんど似たようなものばかりになってしまいました。知っている人なら、一見して、「これProcessingだよね」、「これoFでしょう」と、結局ソフトウェアやフレームワークによって表現が左右されてしまうという状況が、もちろんPhotoshopでも起きてるし、同時にProcessingやoFでも同じ状況だったわけです。

また僕は、元々建築を学んでいて、その当時Grasshopperというモデリングを自動化するツールを使っていたのですが、そのツールも元々は個人、しかも学生がRhinocerosというCADソフトウェアをアップデートしようと勝手に制作し、コミュニティベースで育てていくというものでした。それも数年の後にRhinocerosの開発元のMcNeelに買収されて、オフィシャルのソフトウェアになってしまった。そういうような、オープンソースやコミュニティベースのソフトウェアに対する絶望を学生時代に体験してきたので、やはり草の根でフルスクラッチして戦うぞ、というのはもうかなり厳しいと感じています。そうなるとやはり、これは繰り返しになってしまいますが、そういった相対化されたコードをどこに位置づけるのか、どのソフトウェアとどのソフトウェアを結びつけて、どういうAPIを使って、ある種、ソフトウェアに与えられている社会的な意味を脱臼させていくかみたいなところが、実行的価値になるのかなと思います。そのとき、コードの保存というのは単体ではかなり難しい。なので、僕が書いたコードは、例えば自分亡き後だったり、何年後かに「このシリーズはもう続けなくてもいいかな」と感じた際には、もう実行できなくてもいいと正直思っています。というのも、例えば僕が修復ツールを使うのは、Adobeが2010年に「コンテンツに応じた修復」ツールを実装してわりと話題になったという状況があったからで、そうした社会的状況下でしか成立しえない作品でもあると思っているからです。ある時期にそういう作品が存在した、という事実が残ればいい。これは保存に対する敗北宣言でもあるので、どう捉えるかというのは難しいところですが、ひとまず今の段階ではそのように思います。

久保田　今回、改めてこの「実行」という言葉を出したことで考えたのは、それが環境というか、結局自分がどこで生きていくか、という話と結びついていることでした。思い出したのは、UNIX系のOSでGUI環境を提供するためのX Window Systemをビルドする時に“make world”と最初に入力すること。当時は、このことにすごく痺れました。これから「世界作るぜ!」という感じがあって、世界を今立ち上げる、という感覚の中にいるからこそ、「実行」ということがすごく特別な価値を持ち得たのではないか、ということを今日は再び思い起こしました。だからこそ実行「環境」なのだ、とすれば、やはり保存と記述と実行は、それぞれ独立なのだと思っています。つまり、世界をどうやって保存するかということは、世界そのものであることとは、分けて考えないといけない。それは、やはりメタな関係なのかもしれませんが、そういったことをもう少しうまく整理しながら、引き続き考えていきたいと思いました。