

研究ノート

コードを記述し、実行し、保存する

Writing, Executing, and Recording Code

久保田 晃弘

KUBOTA Akihiro

今日は「コードを記述し、実行し、保存する」というタイトルで、今回のおおがきビエンナーレのテーマとも深く関わるテーマについてのシンポジウムを行います。今日は、僕が今、お話を伺いたいと思っている3人の方々、つまり実際にコードを使って制作をしたり、実践している人たちに来ていただいて、それぞれが「コードを実行する」ということ自体をどう感じているのか、考えているのかということを伺いながら、「コード」とそれを「実行すること」について議論していきたいと思います。

今回の展示に関する資料をGoogle Docs (<https://goo.gl/ze8XGC>) に上げました (99頁参照)。

「コード」の定義

最初にまずお話ししておきたいのは「コードを記述する」と言う場合の「コード」ということばの定義です。コードには広い意味があります。符号もコードですし、いろいろな人間社会のコミュニケーションの中でコードが使われています。だから、コードとは何か、ということを出すとキリがないのですが、今回の議論における「コード」は、基本的に「コンピュータ・プログラム」に限りたいと思います。つまり、コンピュータのプログラム・コードを実際に記述する、実行する、そして保存するということを軸にして、そこから派生するいろいろな状況や現象も含めて議論していきたいと思います。コードにもいろいろあるよね、というような話をしてしまうと、何が何だかわからなくなってしまうので、それはやりません。

「実行的価値」とはなにか

さて、そのことを踏まえた上で、「実行する」とはどのようなことを考えていきます。レフ・マノヴィッチが言うように、僕らの周りで今最もありふれたメディアがソフトウェアです。スマートフォンのようなデジタル機

器はすべてソフトウェアやアプリケーションで動いていますし、それらを実行させることで、僕らはメールでテキストや画像をやり取りしたり、音楽を聴いたり、あるいは映像を観たり、ショート・メッセージを交換しています。つまり日常生活の中で、実行するということは、実はものすごくあたりまえに起こっています。それにも関わらず、実行するということの、特に美術美学的な意味や価値については、今まであまり議論されてはきませんでした。

そこから「実行的価値」という言葉が浮かび上がって来ました。これは例えばベンヤミンが言うところの「礼拝的価値」や「展示的価値」、つまりオリジナルや複製物が持つ価値の先にある価値です。つまり、ソフトウェアを実行したという時に起こっていることから生まれる、つまりソフトウェアを使用することで、それを見たり聞いたりしている人たちが、一体どのような意味や価値を感じているのか。さらに、その「実行」ということ自体が、もし何か美術の文脈において特別な価値を持ち得るとすれば、それは既存の価値と言われているものとどこが同じでどこが違うのか。その議論を掘り下げて行きたい、と思ったのが今日の出発点でした。

具体的には、コンピューター・プログラムを記述をするのは、プログラム言語になります。そして実行することはよく「生成」とも言います。いわゆる「generative design」や「generative art」と呼ばれているもの、それは基本的にはプログラムを実行することから生まれる表現のことを言います。そして保存、プログラム・コード書く人は、常にコマンドS (保存キー) を押そうとしますが、単にコマンドSを押すだけではなくて、実際にコードそのものを今回のテーマであるアーカイブにどう接続・着地していけばいいのか。それらを合わせて、今回は「コードを記述する、コードを実行する、コードを保存する」というタイトルになりました。



図1 久保田晃弘《心をもつ言語》(2017年) IAMAS附属図書館での展示

《心をもつ言語》について

時間も限られているので、今回展示している作品の中からひとつだけ紹介したいと思います。今回の3つの新作の説明のうちのいちばん最初のもの、《心をもつ言語》というタイトルの作品です。この作品は詩です [図1]。そしてこの詩は実行できる詩、つまり「実行詩」です。いま見ているこの画面は、Atomというテキストエディターで、上部のウィンドウに書かれているものが、この「実行詩」そのものにあたります。この実行詩は、全部で5行ありますが、プログラム・コードでもあります。これを今回制作した、実行詩のためのプログラム言語で解釈実行すると、下のShellウィンドウにその実行結果が表示されます。試しに一度、最初から再実行してみます。まず最初に、詩を読み上げます。今回の詩の内容は、この実行詩のコンセプトを伝えるために「Hello, world!」という、プログラミングを学んだものであれば誰もが知っているフレーズとそのバリエーションです。最後の1行だけ、この場にちなんで「Hello, IAMAS!」にしました。この言語の具体的な仕様は、先ほどの資料に書いています。たとえば「H」というコマンドは「Hello, world!」を出力する」ですし、他にも自分自身を出力したり、文字数を数えたり、あるいは、Jenny Holzerの「Truisms」の1節を出力するなど、そうしたいいくつかのことを実行してくれる、非常に簡単な言語です。

詩を実行することで、このように何かを出力してくれるわけですが、それ以外にも、カンマだとかピリオドな

どが、ジャンプなどの制御コードだったり自分を書き換えるコマンドになっていて、詩自身が自分で自分を書き換えたり、あるいは始めに戻ったり、どこかにランダムに飛んでいくことができます。この詩の中には、実行場所を示すカウンターがあって、それ自体を操作したり、その場所にあるコードを書き換えたりしています。

なぜこのような作品をつくったかという、このシンポジウムのテーマである、コードの記述、実行、そして保存ということを具体的に考えるためでした。これらの関係を、まずは簡単な図にしてみました [図2]。

一番左の記述とは何か。この作品の例で言うと、まず上側のウィンドウに詩を書きます。詩というのは、何らかの形式を持っています。それだけでなく、この詩はプログラムコードでもあるので、それを実行することもできます。すると何らかの出力、つまり結果が出てきます。さらに面白いのは、実行すると、外部に一方的に出力するだけでなく、自分自身を読み出したり、書き換えることができることです。つまり「自己参照」「自己言及」が可能になります。するとここから、さらにいろいろな意味が生まれます。ダグラス・ホフスタッターが『ゲーデル、エッシャー、バッハ』で言うところの、ストレンジ・ループです。さらに先ほどの下側のウィンドウに結果が出力されると、それがまた何か別の意味をもたらし表現装置になります。先ほど言ったように、最初のプログラムコード（詩）だけでなく、それらをすべて記録、保存しなければいけません。



図2 久保田晃弘による「記述」「実行」「保存」の図解

そうした時に、アーカイブから逆算してみると、いったい何がどこまでが作品なのか。あるいは「保存する」と言った時に、この点線で囲まれた実行環境のようなのはどうすればいいのか。僕は、このどこを面白いと思ったり、意味を見出したり、あるいは美しいと思ったりしてるのだろうか、ということも議論したいと思っています。今日はちょっとヒアリング的なところもあるのですが、実際にこうしたプログラム・コード、計算機の言語で設計したり、制作をしたりしている人たちが、具体的に何を考え、何をどう思っているのかを、まずは伺いたいと思っています。そこをひとつひとつリストアップしていくことで、これからの議論の、最初の見取り図と言ったら大げさになるかもしれませんが、いわゆるマップのようなものがないかと思っていて、それを今日のシンポジウムのひとつのゴールとして考えています。

いずれにしても、「作品」と言った時に難しいのは、この実行環境を作ることも作品なのか、実行結果やログ

も作品なのか。さらに先ほど実行した詩を見ていただければわかるように、この詩は実行中にとんどん自分の姿を変えていきます。つまり、この詩自体が常に変化しているので、どこでスナップショットをとればいいのか。最終的に、このプログラム・コードはどこかで停止するか、無限ループに入る訳ですが、その時のスナップショットが保存か、というところでもなくて、詩自身が常に変化している場合には、その記録を全部取っておかなければなりません。となると、ますます作品の境界が不明瞭になっていきます。最後に残ったログのファイルや、この詩の形態そのもの、あるいはこの実行する言語のコードのいずれもが作品だといえますが、一方でそれらはどれも不完全なので、単独では作品とはいえないところがあります。こうした形式の問題も、それが意味と不可分なので、重要だと思います。

「実行的価値」とはなにか

まずひとつは「開かれている」価値だと思っています。

具体的に言えば、今この画面上でプログラムが実行しているわけですが、スクリーンショットのムービーを撮って再生した時と実際にコードを実行している時との違い、ということになります。いったん記録してしまうと、基本的には閉じられたものになって、毎回同じものが再生できるようになります。展示の時にも、設営が完了すれば、あとはムービーのループ再生だけすればいいので、すごく心穏やかに展示ができます。しかし外部からの要因や、確率的な要因を含む開かれた実行の場合は、事前にどのような動きをするのか、あるいはどのような出力をするのかはわからない。想定外の入力があれば、バグでなくとも、プログラムが止まってしまうかもしれません。子供が道端で遊んでいる時のように、どんなトラブルがあるかわからないと思うと、何となく不安で、目が離せなくなる。開かれている、というのはそういうことで、実際におこることが、事前に完全には予測できない。未完だからこそ、あるいは失敗する可能性があるからこそ想像できる余白、といってもいいかもしれません。それは、完成すると失われてしまうものでもあります。

もちろんこれは今、言語化して非常にわかりやすくしてしまった「実行的価値」なのですが、実行することによって、それがいろいろ他のこととも結びついていきま

す。それはもちろん、かつてアウラと呼ばれた礼拝的価値や、美術館の展示的価値とも違う、そのあたりを少しでも明確に言えるようにしていきたい、と思っています。もちろんその前提として、レフ・マノヴィッチが指摘したニューメディア、つまりデジタル・メディアの5つの原理、「数による表象」「モジュール性」「自動化」「可変性」「トランス・コーディング」があるわけですが、ここで挙げた自動化や可変性が生み出す価値も含まれると思っています。さらに今夜、東京の神楽坂で「Algorave」のイベントが行われますが、そこで行われるライブ・コーディングのような、リアルタイムにコードを書き換えながら、実行するパフォーマンスの価値とも関連があります。先ほど指摘した、プログラムコードの自己言及性に、さらに人間が介入した時に、この「実行的価値」というのはどう変化していくのか。もちろん「実行とは機械によるパフォーマンスである」と言ってしまうことも可能かもしれませんが、赤羽さんによる《みんなが好きな給食のおまんじゅう》のアーカイブのようなものもありますが、どうも、それとも少し違うような感じがしています。その違いこそに、実行的価値が生み出す、開かれた未完の価値があるのでは、と思っています。