

研究ノート

エンジニアリングとしてのデザイン

Design as Engineering

市川 創太(建築家／doubleNegatives Architecture Ltd.)
ICHIKAWA Sota (doubleNegatives Architecture Ltd.)

プログラミングを多用した建築設計と都市の解析

私が建築士にも関わらず、今回お声掛けしていただいている理由は、実務でプログラミングを多用していることであると想像します。久保田さんとは、私が多摩美術大学で講師をさせていただいていたころからお付き合いがあり、過去に作品で協働もしています。

シンポジウムのお話をいただいて、松井さん、久保田さん、永田さんに事務所にお越しいただき、最初の打ち合わせを持ちました。楽しく雑談をしてその場は終わりましたが、先程、図書館で久保田さんの作品を拝見し、お話を聴き、永田さんのプレゼンテーションを聞きたいま、本日ここでの議論に乗るような話題が提供できるか、心配になっています。

私がマイノリティなことをしているのは自覚していますので、この場で私の立ち位置をクリアにするために、日頃どのような活動を行っているのかということをお話しします。

私の主な活動基盤は、doubleNegatives Architecture (ダブルネガティヴスアーキテクチャー) という緩やかなチームのようなもので、長い活動を経て、現在は建築設計事務所にもなっています。基本的にはクライアントの要望に応えながら、建築設計をしています。もうひとつ基盤として、5年ほど前に勉強会から始まり、現在3人(新井崇俊、國廣純子)で活動している都市研究室 hclab. (エイチシーラボ) があります。建築につながりますが、都市の解析をするためのソフトウェア開発を業務にしています。例えば、商工会議所からの依頼に応じて、調査やポテンシャル評価を行ったり、ベンチャー企業がデリバリーシステムを始めるにあたって、どのようなシステムが適しているかといったコンサルティングをする等、意識的に現実の都市へのアプリケーションを課題としています。

コードによるコラボレーション

アーティストとのコラボレーションもしています。三上晴子さんと《gravicells》というピースを制作しました [図1]。平面6mx6m程度のインタラクティブなインスタレーションです。作品のサイズとしては比較的大きく、私は建築的なスケールで、その体験がどのようにありえるかを考えながら制作しました。本作品のコードは、大部分を私が書いています。

中谷芙二子さんとも一緒に作品をさせていただいています。私がいまでもなく霧の彫刻家として著名で、フランスの芸術文化勲章の最上位を受勲された作家です。作品だけでなく、DVD-ROMのインターフェースのデザインやソフトウェア、データベースのプログラムも担当しました。フランスから出版されている"Anarchive" というアーカイヴシリーズのひとつに中谷さんの"FOG" という作品集があり、その付録のDVD-ROMです。

久保田さんと協働させていただいたのは《The Cellular Automaton Band》というラップトップバンドで、公式サイト (<http://cab.d-xx.com/>) があります。私はバンドブームの世代で楽器を多少やっており、趣味の話が盛り上がりすぎてしまったバンドです。ジェネラティブであり、セルオートマトンのセルをトリガーとして演奏するものです [図2]。

セルオートマトンは、かなり枯れた、やりつくされたもので、ご存知の方も多いと思います。パラメーターによってはセルがたくさん栄えたり、しぼんでいったりという緩やかな操作ができます。そのしぼんでくだろうというパラメーターを演奏の最終盤に設定して、そのままステージを降りてしまうのです。直接手を下さずとも、セルが全部死滅して勝手に演奏は終わる、というような演出です。勝手に演奏が終わる演出のはずが、以前横浜で演奏したときには終わらなかったこともありました。

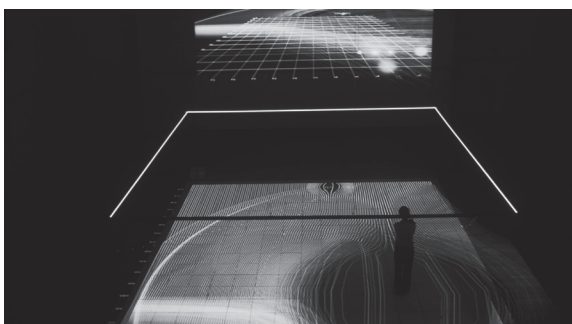


図1 《gravicells》 photo: Hiroki Obara
Courtesy of Yamaguchi Center for Arts and Media [YCAM]

セルがすべて死滅する直前の数ステップで、いくつかのセルが偶然ブリンク状態になってしまったのです。そういった操作しきれない性質は、ジェネラティブな成り立ちならではものと言えます。

《Corpora in Si(gh)te》は、doubleNegatives Architectureで企画・制作したのですが、YCAMのコミッションワークの中でも、かなり大きなインスタレーションでした〔図3〕。いろいろな機会に恵まれて、ヴェネツィア・ビエンナーレに出展したり、メキシコのアルス・エレクトロニカに行ったり、様々な会場を巡回しました。2008年頃の制作ですが、当時あまりユーティライズされていないAR（強化現実）を無理矢理使って、メッシュネットワークセンサーといった技術なども取り入れました。プロジェクトには4人以上のプログラマーがいましたが、コードを共同で書くという作業が非常に新鮮で、当時Gitは一般的でなく、Subversionというソースコード管理システムにTortoiseというクライアントを使いました。

さかのぼると、私は大学院を出てすぐ、Knowbotic Researchのプロジェクトに参加させていただいたり、当時京都で進行中であった荒川修作さんとマドリン・ギンズさんのプロジェクトの実施設計をお手伝いしていました。1990年代は、荒川さんの不思議な魅力、得体の知れなさが、強く印象に残っています。1960年代に渡米されて、その空気感を冷凍保存して日本で解凍したようなイメージです。言説にも強烈なインパクトと、破壊力がありました。「日本には建築家なんかひとりもない」という辛辣な話から入るのは毎回で、それに釣られた私は、二つ返事で参加しました。残念ながら実現せずに終わってしまったプロジェクトです。



図2 《The Cellular Automaton Band》

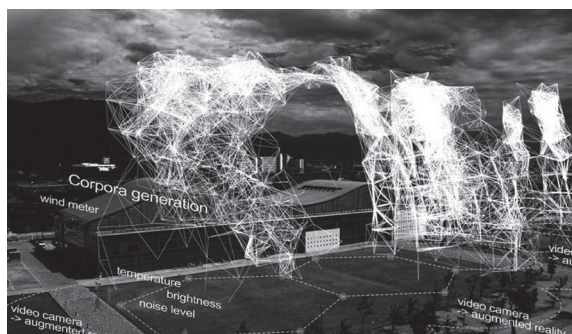


図3 《Corpora in Si(gh)te》
Courtesy of Yamaguchi Center for Arts and Media [YCAM]

道具としてのソフトウェア

今日はコードに焦点を当てた会ですから、なぜコードを書き始めたのか？ という内容を話そうと思います。とはいえ、ことの経緯は個人的なきっかけの連続でありますので、歴史の中の位置づけとしては話しきれません。私は建築科を卒業していますが、美大出身です。自分に近い分野で手の届く範囲には、やろうとしたことを叶えるための既成のソフトウェアがなかった（自分独りでは見つけられなかった）ことで、無いものは作るしかない、という成り行きでコードを書き始めたきっかけのひとつです。

当時CADソフトが建築設計事務所に導入されてきて、図面の描き方が変わる過渡期でした。手描き作図もCAD作図も両方経験している世代です。CADモンキーと揶揄されたカテゴライズに入るほどCADばかり使って図面を描いていました。次第に、ソフトに使われてる、ソフトの設計者の意図に従わざるを得ない、ソフトの特色とアウトプットの特色が紐づいて見えてしまう、と感じる経験をしました。現在は、ソフトウェアが多様化していますので、ソフトの特色がアウトプットに染みついてし

まうようなことはないかもしれません。当時は、「あ、これはあのソフトで作ってるんだろうな」というような印象がアウトプットから匂ってしまうようなところがあり、道具としてのソフトウェアにクリエイティビティが左右されてしまうのではないかという疑問がありました。

表記方法、図法から見出される表現

建築の教育を受けると、専門的な図法や表記方法を読み書きできるようになります。建築家の図面や表記方法に対して、例えば音楽家の楽譜と表記方法、これを比較して考えてみましょう。西洋音楽のコンテキストから、楽譜に必要なとされた主な機能のひとつに、讃美歌の流布があるようです。それは教会でミサ曲を信者に覚えさせるということです。「グイードの左手」というのが、初源的な楽譜と言われているらしく、曲中の繰り返しのリフレインなどを左手に記してある様子が、挿絵から見られます。曲のパターンを記述して、覚え伝えることから、音を記録する、音を記述するということです。耳コピーと言われるように、耳で聴いて演奏を再現できますが、記述することで音楽を正確に伝える方法が発明されたいうことだと理解しています。

ここまでは記録が出発点ですが、楽譜は、作曲家が作曲を行う上で、楽器を使って演奏しなくても、楽譜上で試行錯誤をしたり、シミュレートができる。思考を補助する道具として使われてきたのではないかと考えてみました。バッハの《フーガの技法》という作品集の中に、楽譜を上下反転させても成り立つ曲、楽譜の後ろから演奏しても成り立つ曲があります。バッハは、非常に高い才能とスキルをもっていたと想像できるので、彼の頭の中だけで、殆ど曲を構築できたのかもしれません。一方で、このような表記、記譜を通してビジュアルライズしたことをきっかけに、図形的に反転したり、逆さまに読むなどの発想に至る可能性も想像できます。音のシークエンスが形になり、ある表記方法によって写されたことで、目で見たと形のパターンを逆転するというように、表記方法があるからこそ可能な発想があるのではないかと考え始めました。

建築に話を戻すと、建築を全てひとりで行う。セルフビルドしてしまうこともできますし、居住できる洞窟を見つけることも初源的な建築という行為になりそうです。ただし、社会の中で行うとなると、決められたルー

ルに適応させることも必要です。大きなものを多くの人が協働して作る際、同じ完成物を目指すために、不足なく準備するためにも、建築の計画を記述して共有する必要がありますがでできます。

建築のスケッチには、紙の端切れにメモを書いたり、パースを描いたりすることがあると思います。そのメモ程度のフリーハンドの絵は、時に透視図だったり断面図であったり平面図であったりします。表記方法（図法）があったからこそ、表出した発想なのか、発想を図法にしたがって記述しているのか、どちらが先とも言い難い。設計で日常的に使っている平面図、立面図などと異なった空間表記方法を作って、その上で考えたり描いたりすると、従来とは異なった空間概念に到達できるのではないか。この問いが私の活動の出発点になっています。

《Super Eye (Smooth compound Eyes)》

私が空間表記方法として提案したものはパノラマビューを想像するとわかりやすいものです [図4]。水平方向360度鉛直方向±90の角度を方眼展開して、表記対象物までの距離情報（深さ情報）を埋め込みます。二つの角度と距離で極座標として3Dの情報が確保できるので、この方法をベースにした空間表記方法を提案しました。当初エクセルという表計算ソフトで、座標を変換計算して、その結果をひとつひとつ方眼紙に記すことで実現しました。表記対象によっては、量的に無理ということで、当時Turbo C、Turbo C++といった安価な開発入門ソフトを買って始めたというのが、私のコーディングの始まりです。

《Corpora (collection(s) of Super Eyes)》

自分にとってはあくまでも建築をどのように設計するかということが主題でしたが、基点＝視点を同時に複数存在させることで、鳥の群れが飛行する状態の記述を試みたり、時間をかけていくつかのステージを踏みながら表記方法の応用を考えていきました。この表記方法は一つの基点から空間を計測する極座標が基本の座標原理になります。これを建築構造の結節点に基点を置くことで、構造部材がどのようにつながっているかが把握でき、結節点を局所的にコントロールすることが可能になると考えました [図5]。

この方法と考え方を建築にアプライして解を探っていく過程では、結果が大量に出てきます。その結果群に俯

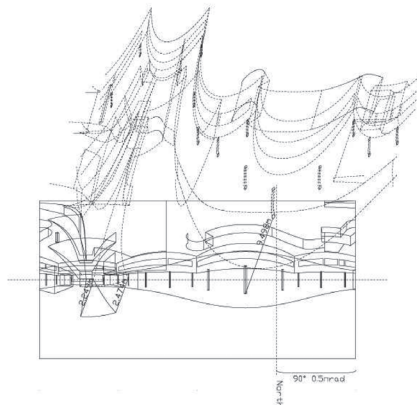
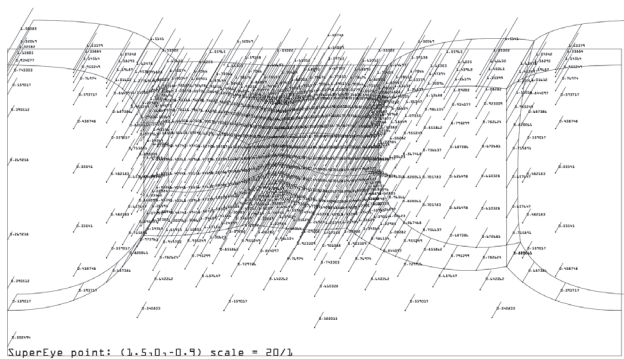


图4 《Super Eye (Smooth compound Eyes)》

瞰的なフィルターをかけていきます。居住できるようなボリューム無い、構造として部材数が少なすぎる、部材の接続が多すぎて施工性が悪い、などをフィルターで取り除きながら形をさぐるというような応用を行いました。《House in Nagohara》はそのひとつの結果です [図6]。

《HUGO 羽後街道の家》

これも建築の実施プロジェクトです [図7]。コストが比較的厳しかったのですが、それをカバーできるクライアントの理解と思い切りがあり、設計を引き受けました。この住宅は東北で積雪のある地域を敷地としています。

敷地内の除雪作業を軽減するために、敷地内の通路を短く、できるだけ道に接することをひとつの条件にしました。予算的に屋根の形は4本の直線で囲ったシンプルなものを2つ目の条件に。コアと屋根の重心がずれてると構造的なバランスが悪く無理部材が大きくなるので、コアと屋根の重心を一致させることを3つ目の条件にしました。その3つを満たす形を導く数式が作れます。

建物の配置角度によって、屋根面積（建築面積）がどれ程になり、冬に太陽光が入り、夏は直射日光が室内に入りにくい向き、日射に関連した軒の出具合、それらの関係やトレードを把握し、無数のコンビネーションのな

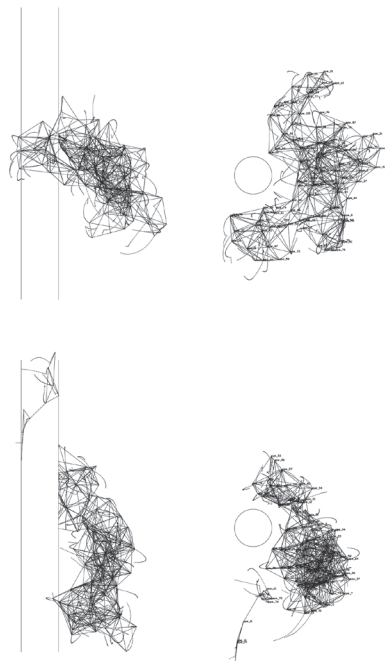


图5 《Corpora (collection(s) of Super Eyes)》

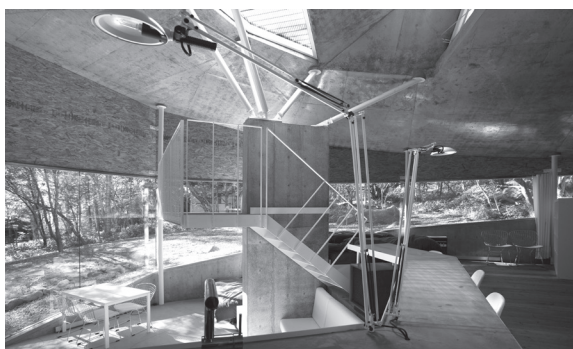


図6 《House in Nagohara》 photo: Kenta Ichikawa

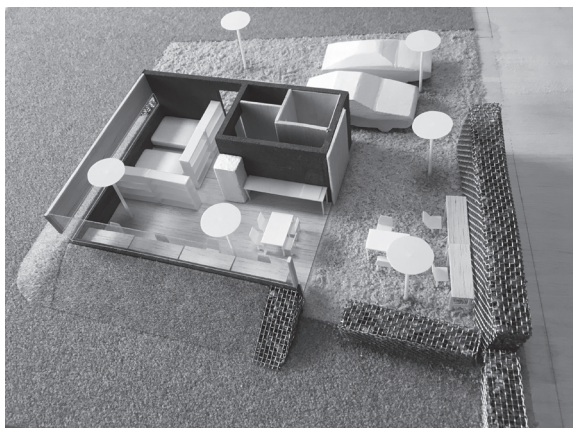


図7 《HUGO 羽後街道の家》

かから好ましい解を見つけ出す為にソフトを開発しました。このプロジェクトでは、1年間の太陽の、10分毎の高度と方位の情報を埋め込むための評価点として、敷地内に約十万点を設定しています。評価点を利用して、建物の配置や屋根の形状、軒の出などによって変化する建物内への日射の分布や量を統計的に見ていきます。

諸条件により、屋根形状を決めた後、柱の位置を別のプロセスで決めました。ゲームなどに使われている物理エンジンを援用しています。中央のコアは、初期与条件としてスタティックにし、柱を配置しない場合、あるいは柱の置き方によってはたわみが出るモデルを作り、柱をひとつずつ配置可能なグリッドの交点へ動かしていきます。柱自身が適切な場所を探しに行くようにも見えます。たわみの大きいところほど色が赤くなりますので、ヒートマップとして全体が青くなる柱の配置を探していきます。古典的なシミュレーテッド・アニーリング（焼きなまし法）という手法を使っています。

建築設計で解くべき問題設定を考え、デザインして、

それを解くためにコンピューティングをどのように使うかを日々考えています。コンピューティングを使った建築はたくさんありますが、どのように設計の問題を抽出し、手法をアプライするか、そのやり方は依然出きっていないのではないか、と私は考えています。未だに興味とモチベーションを持って取り組んでいます。

「時間地図」のデジタイズ

私は開発にQtというC++用のフレームワークを好んで使います。

クロスプラットフォームで良くマニュアルも整備されていることがひとつの理由です。スクリプト言語か、ハードコードか、開発の初期で様々な分かれ道がありますが、ハードコードをしながら、処理速度が出せるような開発環境と実行環境を選んでいきます。都市コンサルティングなどの業務では、計算の速度や量が必要になってくる場合が多いからです。

こちらの例は、上野エリアを対象にしたものです〔図8〕。都市の中の移動は連続平面だけでは捉えきれません。街路があり、私有地の敷地を突っ切って目的地に行くことはできません。単なる点と点を結んだ距離で、都市の中の距離を測ることは正確ではありません。ある地点からの徒歩で到達できる時間マップを描くと、表示のように実際の地図とは異なってきます。街路ネットワークの上では、まっすぐに進めない場合がほとんどなので、実際の地図よりも出発点から外に広がってしまいます。このソフトは、ネットワーク内の最短距離を瞬時に計算できるようになっており、この例では、5分毎の到達時間の範囲を示す同心円が見えます。10分圏内、20分圏内ということが一目瞭然となります。1970年代にグラフィックデザイナーの杉浦康平さんが、時間によって地図を描き変えるという「時間地図」を考案しました。杉浦さんのマスターピースをhclab.でデジタイズして再現したということでもあります。

このソフトの入力情報として、公共交通移動を考慮したネットワークを入れると時間地図も変わります。ある街路を徒歩よりも高速で移動することは、移動時間が短くなることであり、2点間の距離が短くなるように近くに引き寄せることである、ともいえます。公共交通が、都市の中の各点を引っばっている様子を見ることが出来ます。出発点は、自由に変更でき、指定した地点から、瞬時に時間地図が描けます。強く引かれている2点間ほ

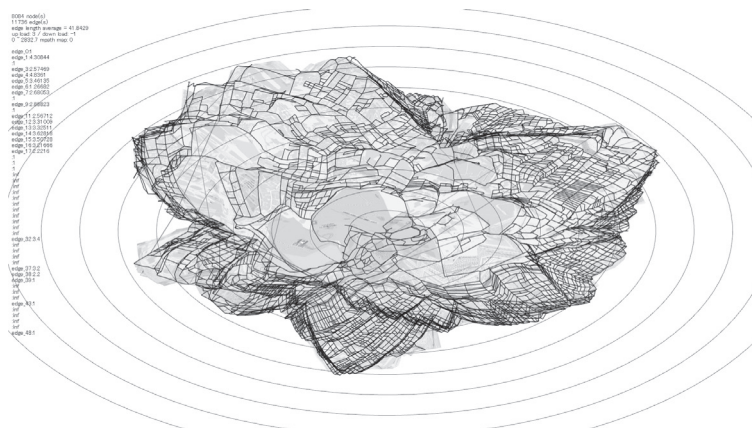


図8 hclab.による上野エリアのサーベイ

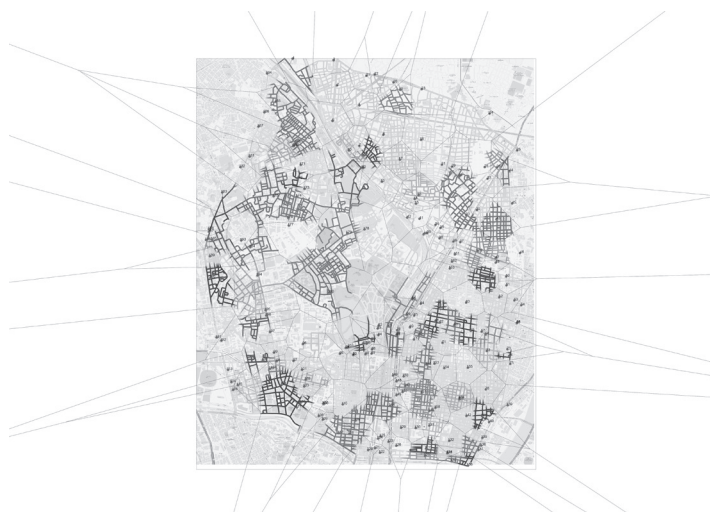


図9 ネットワーク上のボロノイ図

ど、運行頻度あるいは速度の速い交通で運ばれている、と見るができます。

エンジニアリングとしてのデザイン

このような技術を基本にして、様々な計算機能が必要に応じて追加しています。

例えば、青色で示したいいくつかの地点に行くのに利便性のよい場所を、計算で導くことができます。都市の居住者達にとって、平均的に近いということや、最も不便なところ、いちばん遠く、条件の悪い地点の居住者までの距離を最小限にできる地点はどこなのか、ということ把握することができます。利便性のよい商店街はどこに作るとよいか、不便な地点に住む人も取りこぼさないための救急車のステーションがどこあるべきか、という

ことを科学的に検討することができます。

都市や建築規模で何かを行うには、非常に大きな投資が必要になるので、ある程度計算に基づく事前の把握がデザインの下敷きとして必要です。デザインを支援する目的でソフトを作り、ソフトウェアを随時カスタマイズしてコンサルティング業務に使用しています。

次の例は、画面で指定した地点の、最寄りのエリアを計算します [図9]。連続平面では、ボロノイ図として知られていますが、ネットワーク上でのボロノイ図は、結果や見た目が異なります。この例では、桃色の領域がボロノイ図で想像される形よりも歪んでいて、母点から遠くの領域に及んでいます。この母点からの移動に際し、公共交通のネットワークが周辺領域をカバーしているためです。次の例は、指定した地点同志を、最小限の枝、

つまり街路でつないだミニマム・スパニング・ツリー（最小木）がつくれます。指定した各地点に商店があるとすると、それらが最小限の道路でつながれたネットワークであるのならば、効率のよい商店街として考えられます。このように案件に応じてモデルや評価方法を検討し、ソフトをカスタマイズしながらコンサルティング業務を行っています。

同じ理論をベースにしている、応用先はひとつではありません。

こちらは建築設計の問題を考えたものの例です。露出鋼管での電気配線は、材料と手間がかかります。建物内の電気配線の大部分を露出配管で行うことが条件の場合、管中の電線ケーブルが長くなることを許すと、鋼管の総長をミニマムにすることはコスト減につながります。この配管の形に、前述の最小木を応用することができます。あるいは、いびつな居室平面に、照明器具をどのように配置すると光がうまくディストリビュートするか、これも小さな問題ではありますが、形に現れてくる要素です。それらを感覚や経験値だけで決めていくのではなくて、計算と妥当な評価をしながら決めていく。デザインがエンジニアリングとしてなされるべきだと、強く思います。

コードを機能から評価する

コードの実行や、コードそのものの位置づけに対して、結論を導くのは難しいですが、私には、コード自体がその実装機能と切り離して、何か価値があるとは考えにくい。動作するかしないか、コード自体の価値というよりは、それがどのように使えるか、という判断が尺度になります。コードを理解し、アプライする。組み合わせによって可能になることを想像する。複雑に見える現象を一元的に記述する。といった使い方や、アイディアの方が重要です。私がメディア・アートに関わらせていただいていることや、美大出身ということもあって、コードの美学のようなものを期待していただいたと想像しますが、おそらくコードそのものに対しては、ぞんざいな扱いをしています。コード自体を台座に座らせてるような扱いはしていません。

コードの実行的価値ということは、本企画開始当初からお話がありましたので、継続的に考えてはいます。久保田さんが仰るような感覚は、決して想像がつかないということではなく、プログラミングしてる人にしか得ら

れないような感覚、ミュージシャンズ・ハイのような、演奏中のある種の恍惚感のようなものと言えるかもしれません。時計職人が、動作を確認するときに感じるであろうプレッシャーとプレジャーが共存するような感触も多分にあると思います。

質疑応答

久保田 具体的な例で言うと、コードはどのように保存しているのですか？

市川 昨今はソースコードのメンテナンス技術が進み、さまざまなシステムが開発されています。にも関わらず、私の保守管理はずさんで、プロジェクトを丸ごと消失させたり、驚くようなミスをしています。前述のセルオートマトンバンドのソースコードはすべて無くなっていました……。ソースコードの保守も、プログラミング技術の一部と認識しますが、かなり異なった知識や技術を必要とするのではないのでしょうか？

Subversionというバージョン管理システムの上で複数人で開発した経験があります。よくできた管理システムではありましたが、混乱しました。インсталレーションのための開発で、締切直前の現場にも関わらず、バージョンをいくつも戻すような場面が多々あったように記憶しています。現場での開発の混乱や、プログラマー同志のコードの混同は、すごく焦りますし困るのですが、勉強にはなります。差し迫った現場で「勉強」といった悠長なことは言ってもらえませんが、プログラミングというものは、特に勉強が必要だと思いますね。

久保田 バッハの例がありましたが、作曲と演奏というものがあつたときに、それは建築設計と建設の関係と同じだと思っていいのでしょうか？

市川 計画までが建築設計の主な仕事であると思う節があります。現場に行ってチェックして、設計の通り施工が行われているかを監理する、適合する素材を探してくるという設計者の役割は当然ありますが、建築は計画のときにこそ、ドラスティックに変更できます。最も方向分岐が多い、計画の時点で使われる言語が、平面図などの空間表記方法ではないかという問い、思考が言語に左右され、発想が言語に縛られる、という疑問につながります。