修士論文 2011 年度（平成 23 年度）
Master's Thesis Academic Year 2011

アルゴリズミックにデザインすることの優位性に関する考察

～デザイン課題への応用経験を通して～

A study on an Advantage of Algorithmic Design Approach : Through an experience of application to Design Projects.

慶應義塾大学 政策・メディア研究科
Keio University, Graduate School of Media and Governance

阿部祐一
U-ichi Abe

アルゴリズミックにデザインすることの優位性に関する考察

～デザイン課題への応用経験を通して～

論文要旨

現代におけるデジタルデータ利用普及とコンピュータの進歩は、人間の生活と社会に大きな変化をもたらした。多くの情報がデジタルデータによって表現可能になり、それを高速で処理するコンピュータの役割は、今や人間の生活と社会にとって必要不可欠なものとなった。

デザインを取り巻く環境も、こうした現状と無関係ではない。特に、デザインプロセスにコンピュータを取り入れる上で、アルゴリズミックにデザインするということの必要性と重要性が認識され始めた。

本研究では、こうしたアルゴリズミックにデザインするということが果たしてどのようなことであり、そして既存のデザインに対してどのような優位性を持つのかということを考察する。

研究の手法は、著者が関わったアルゴリズミックなデザイン事例と、その経験を通して著者が得た知見を分析することで、アルゴリズミックにデザインするということの本質に近づけるのではないかと考えた。

分析の対象となったデザイン事例は、個々に独立したデザイン課題を持つものであったが、その全てに著者が関わっているということから、個別のケーススタディであるとともに、アルゴリズミックなデザインを深化させていく過程であるともいえた。

こうした具体的なアルゴリズミックなデザイン事例と、それに直接関わったという著者の経験から、アルゴリズミックなデザインを適切に運用ための多くの有用な知見が得られ、またアルゴリズミックにデザインすることの優位性についてを考察することができた。。

アルゴリズミックにデザインすることは、デザインにおける協働の可能性を大きく広げるものである。アルゴリズミックにデザインが用いられる理由の多くが、人とコンピュータが協働することで、人だけでも、コンピュータだけでも解決が困難なデザイン課題を解決するためである。しかし、適切に運用されたアルゴリズミックなデザインでは、通常のデザインでは、デザインの外側に置かれてしまう " 他者 " がデザインに参加することを可能にする。

またアルゴリズミックにデザインすることは、適応的なデザイン、プロセスのデザインを可能にする。これは今までのデザインが踏み込むことができなかった、新しいデザイン領域である。

キーワード

1. アルゴリズミック・デザイン　2. コンピューテーショナル・デザイン　3. 都市デザイン
4. 自己組織化　5. シミュレーション

慶應義塾大学 政策・メディア研究科

阿部祐一

A study on an Advantage of Algorithmic Design Apploach

: Through an experience of applications to Design Projects.

Summary

The widespread use of a digital data and the development of a computer changed a human life and society.
Now, we can describe many information in a digital data. A computer that hundl these digital data at a fast become an essential tool for human life and human society.
These situation is not only around a human life and human society but also around a design. Especially, we take note of a necessity and inportance of algorithmic design to introduce a computer to a design .

In this study, I look at what is algorithmic design approach and what advantage have algorithmic design approach compare with another design approach.
analyzing a case study of algorithmic design I assisted and my experiments of these, I thought to be able to approach the stuff of algorithmic design.
These design projects have independent design problem. These are not only a independent case study but also deeping process study, because I assisted to all these projects.

I could obtain Many knowledge and information that are are very useful to manage algorithmic design from these case study and my exteriment. And Similary, I could look at an advantage of algorithmic design apploach.
To design with algorithmic method incresethe the possibility of a collaboration in design process.
And to design with algorithmic method realize a adaptive design and enable designing a process.

Key Word
1. Algorithmic Design 2. Computational Design 3. urban Design 4. Self-organization
5.Simulation

Keio University, Graduate School of Media and Governance
U-ichi Abe

目次

1. 序章 -1

1.1. 研究の背景 -1

1.1.1. デジタルデータによるデザインプロセスの普及 -2

1.1.2. コンピュータの計算能力によるデジタルデザインの処理 -3

1.1.3. コンピュータを用いたデザイン -4

1.1.4. コンピュータとアルゴリズム -4

1.1.5. コンピュータによるデザインの模索範囲の拡大 -7

1.2. 研究の目的 -6

1.3. 研究の意義 -6

1.4. 研究の方法と構成 -6

2. アルゴリズミックなデザインによるデザイン経験とその考察 -8

2.1. アルゴリズミックなデザイン事例個々に対する考察 -9

2.1.1. デザイン事例 1 : Alternative Suburban -10

2.1.1.1. デザイン課題の整理 -10

2.1.1.2. デザイン・パラメータ -12

2.1.1.3. デザイン・アルゴリズム -12

2.1.1.4. デザイン・パターン -13

2.1.1.5. 得られた知見 -17

2.1.2. デザイン事例 2 : Porous Torus -19

2.1.2.1. デザイン課題の整理 -20

2.1.2.2. デザイン・パラメータ -22

2.1.2.3. デザイン・アルゴリズム -22

2.1.2.4. デザイン・パターン -23

2.1.2.5. 得られた知見 -27

2.1.3. デザイン事例 3 : ドミエメロード -29

2.1.3.1. デザイン課題の整理 -30

2.1.3.2. デザイン・パラメータ -32

2.1.3.3. デザイン・アルゴリズム -33

2.1.3.4. デザイン・パターン -33

2.1.3.5. 得られた知見 -37

2.1.4. デザイン事例 4 : 大連大和館・別館 -39

2.1.4.1. デザイン課題の整理 -40

2.1.4.2. デザイン・パラメータ -42

2.1.4.3. デザイン・アルゴリズム -42

2.1.4.4. デザイン・パターン -43
2.1.4.5. 得られた知見 -48
2.2. アルゴリズミックなデザイン事例を通じた考察 -50

3. アルゴリズミックな手法による実験的デザイン -53
3.1. デザイン対象の設定 -54
3.2. デザイン対象に対する調査・研究 -54
3.3. デザイン事例からのフィードバック -56
3.4. アルゴリズミックな手法の検討 -56
3.5. 想定される具体的なデザイン -58
3.6. デザイン提案に対する総括 -44

4. 結論 -63
4.1. 総括 -63
4.1.1. アルゴリズミックにデザインするためには -63
4.1.2. アルゴリズミックにデザインすることの優位性 -68
4.2. 今後の展開 -71

1. 序章

1.1. 研究の背景

　　現在デザインの分野において、アルゴリズミックな考え方の重要性が盛んに論じられている。それはデジタルデータ利用の普及とコンピュータの発達が、デザインにおけるコンピュータの役割が増してきているためと考えられる。

　アルゴリズミックな考え方に基づくデザインにおいては、人とコンピュータが協働するような新しいデザインプロセスを構築することができ、それがデザインそのものの可能性を広げることにつながると考えられるからだ。

1.1.1. デジタルデータ利用の普及とデザイン

　現在、人間を取り巻く多くの情報が、デジタルデータとして扱われるようになっている。

　もともと、人間の周囲にある物理空間上の情報は、時間的に連続でかつ値としても連続的に変化するアナログな情報である。こうしたアナログな情報を、離散的な値によって表現したものがデジタルデータと呼ばれるものだ。デジタルデータには、1. 計算による情報処理に向いている　2. 様々な情報を統一のプラットフォームで利用できる　3. 情報の正確な複製、伝達、共有が容易である　4. 情報の劣化を防ぐことができる　5. アナログ情報を完全には再

現不可能である　といった特徴がある。

デジタルデータ利用の普及は、デジタルコンピュータに代表される計算機の進歩、普及と密接に関係している。しかしその関係は必ずしも、デジタルデータ利用の普及が計算機の進歩、普及によるという従属的なものではなく、デジタルデータそのものの利点が計算機の進歩、普及によって生かせるようになったという相互的なものであると考えられる。

ともかく、デジタルデータの利用手法、またはデジタルデータによる表現手法の進歩により、デジタルデータの精度、処理能力、利用規模などが向上したことが、デジタルデータ利用の普及に貢献し、一般化してきたといえる。

もちろんデザインを取り巻く環境も、こうした流れに無関係ではない。デザインに関係する多くの情報、地理情報、空間情報、物理情報、図形情報などがデジタルデータ化されるようになったことに伴い、デザインのプロセスにもデジタルデータが多く活用されるようになってきている。

1.1.2. コンピュータの計算能力によるデジタルデータの処理

デジタルデータは計算機による情報処理と大変相性がよく、計算機の進歩、普及がデジタルデータ利用の普及と無関係でないことはすでに述べたとおりである。

計算機とは、現在ではデジタルコンピュータと呼ばれる電子回路を用いた、高い計算能力を持った機械を指すことが一般的である。しかしその起源は、単純な数を記録する手法や道具に始まり、古くは、手動操作を伴う計算尺やそろばんなどの計算器具、そして歯車を用いた機械式計算機、そして物理現象を用いたアナログコンピュータなどを指す言葉でもあった。

こうした時代から、計算機は人間の活動を支援する強力な機械であった。デザインの分野で言っても、建築物設計における構造計算や、航空機設計においける航空力学的計算など、物性や物理法則の計算などにおいて初期の計算機は大きな力を発揮してきた。そして、デジタルデータ利用とデジタルコンピュータの開発が、こうした計算機の有効性を大幅に拡大することとなる。

前述したデジタルデータ利用の普及が、コンピュータが計算処理可能な情報の範囲を拡大させ、コンピュータの計算能力向上は、計算の精度、扱えるデジタルデータの規模、デジタルデータ処理の速度を飛躍的に上昇させた。また、コンピュータの普及と製造技術の進歩から、個人が十分な計算能力を持ったデジタルコンピュータを安価に入手できるようになり、コンピュータ利用の浸透から、複雑高度な計算処理を自動化してくれるソフトウェアの発達と普及も進んだ。そして、こうしたコンピュータとデジタルデータの進歩が、さらに両者の利用の普及を加速させ、進歩そのものも加速させるという正の連鎖を生み出し、両者は爆発的な発展を遂げることとなった。

1.1.3. コンピュータを用いた新しいデザイン

デザインの分野におけるデジタルデータとコンピュータの利用は、人間のデザイン行為を支援する目的で発達してきた。

初期において計算機は、デザインに関わる物理的情報や物性を計算することに使用された。建築デザインにおける構造計算に、かつては計算尺が多用され、それが現在ではコンピュータや電卓によるものに取って代わられている。コンピュータの計算速度が向上することで、高度な構造計算がコンピュータ上のソフトウェアとして実装されることで一般化し、かつては解析が困難であった複雑な構造体の構造解析が可能になった。

CG(Computer Graphics) も、デザインの分野に大きな影響を与えたデジタルデータとコンピュータの成果である。CG は、コンピュータで処理された情報を画像という視覚情報に変換・出力したもので、デザインにおいてコンピュータによる情報処理と人間の感覚・感性を繋ぐという重要な役割を果たしてきた。現在では、多くのデザイン成果が CG によって作成されるようになっている。

コンピュータによってデジタルデータ化された写真などの 2 次元画像や、それをコンピュータで処理をした CG は 2DCG と呼ばれる。こうした 2DCG を扱うコンピュータソフトウェアとしては Adobe の Photoshop が有名であるが、これによる画像の修正・加工は、多くのデザイン分野において必要不可欠なものになりつつある。

また、3 次元空間を平面上で表現した画像を作成するレンダリングという技術と、それによる 3DCG もデザインの分野では多用されている。現在の 3DCG は、写真と区別がつかないほど写実的な画像を作成することや、高速で 3 次元空間を描画することでコンピュータ上で仮想的に 3 次元空間を再現することも可能になっている。

コンピュータによる図形情報処理は工業分野で応用され、複雑な図面データを高精度かつ容易に作成、共有する手法、CAD(Computer Assisted Drafting) となった。CAD は建築などのより大規模な設計への利用も普及し、現在では 2 次元的な図面データだけでなく、3 次元的なモデルを仮想空間上で自由に扱うことのできる 3DCAD が一般化してきている。

また、通常は別々に作成、運用されるデータを 1 つに統合して作成・運用する新しい建築デザインの形、BIM(Building Information Modeling) というものも登場した。建築に関わる情報をデジタルデータによって統合し、容易に共有することは、複数の主体によるデザイン協働とインタラクションを生み出し、製造や施工、コストなどを含めた建築全体のライフサイクルをマネジメントすることを可能とする。

製造業の分野では CAM(Computer Aided Manufacturing) と呼ばれるコンピュータ制御による加工が発達した。CAM による加工機械と技術が成熟し、一般化し始めた現在では、それを前提としたものづくりであるデジタルファブリケーション、パーソナルファブリケーションという分野が、新しいデザインの領域として現れている。

コンピュータによるシミュレーションは、複雑系という新しいシステムの発見と解析にも役立てられた。複雑系は、人間社会、自然現象、生物行動に多く見られる有用なシステムであり、既存の還元主義的アプローチでは解決できない問題に対しその力を発揮する。その解析と利用は、デザインとそのベースとなる科学にまったく新しい視点を持たしている。

このようにコンピュータの利用は、デザインに多くの成果と新しい可能性をもたらした。

1.1.4. コンピュータとアルゴリズム

K.Terzidis が「アルゴリズミック・アーキテクチャ」において述べているように、アルゴリズムとは、デザインにおける「思考法」であり、人とコンピュータという異なるシステムを「組み合わせ」ることのできる手法である。

デザインにおけるコンピュータ利用事例において、コンピュータはアルゴリズミックな処理を高速・高精度に自動実行するという役割を与えられている。アルゴリズミックな処理とは、明確に手順化されたプロセス、アルゴリズムに従った処理のことである。アルゴリズミックな活動は人間活動においてもみられるものであり、単純作業や数学的・論理的プロセスなどがそれにあたる。

コンピュータの行う計算処理とは、アルゴリズムに従い、与えられたデジタルデータに一定の処理を加え、その結果を出力するという作業であり、かつその過程が自動的であるという点に大きな意味がある。いコンピュータは、人間のように曖昧性をはらんだヒューリスティックな判断や、感性的な判断ができない代わりに、こうした作業において自身のもつ圧倒的な計算能力を最大限発揮し、人間のデザイン行為を支援している。

例を挙げるならば、コンピュータは 2DCG 画像処理において、CG を構成する画素の色素数値に対してアルゴリズムに基づき、延々と一定の計算処理を行っているだけである。しかし、こうした処理が高速化、自動化されていることで、人間は画像に対してコラージュや拡大・縮小、色調変換などの処理を容易に加えることができる。

3DCG のレンダリングにおいては、3 次元空間における光の軌跡や減衰をトレースし、その結果を 2 次元平面状に投影し、描き出している。これらの処理は数学的に定義された計算をアルゴリズムに従って繰り返しているに過ぎないが、結果として出力された 3DCG は現実の空間と比べても遜色ない程のリアリティを獲得し、その速度は現実空間でのリアライズよりも遥かに高速である。

このようにアルゴリズミックな考え方とは、コンピュータによる計算処理を人がデザインなどの人間活動に取り込む上で、必要不可欠なものである。

1.1.5. コンピュータによるデザインの模索範囲の拡大

C・アレキザンダーは「形の合成に関するノート」で、デザインをある問題を解決するための答えとなる形を求めることと述べた。しかしデザインには、数学のように求解のための明快な式などは存在しないため、数多あるデザインの可能性の中から答えとなるデザイン解を検索、模索する必要がある。

しかし、人によるデザイン解の検索は人間の思考・作業速度の限界から、その検索範囲に限界が生じてしまう。そのため従来のデザイン手法では、人間の経験や感性に頼って、人間が検索可能な範囲まで検索範囲を限定し、その中で検索を行うという手法がとられてきた。

コンピュータによるデザインプロセスの高速化は、こうした人間が検索可能なデザイン解の範囲を大きく広げることとなった。CAD による図面の高速作成、CAM を利用したファストプロトタイピング、コンピュータシミューレーションによる仮想モデルでの検証などである。コンピュータをデザインプロセスに取り込むことは、人間のデザイン検索可能範囲を拡張するという可能性を示唆している。

しかし、デザイン解の検索可能範囲を闇雲に広げることにあまり意味がない。あくまでもコンピュータによって拡張されたデザインの検索可能範囲が、人間に扱うことのできるものであること、そしてその拡張がデザイン解の向上につながることが条件である。

このように、デザインにおけるコンピュータの利用では、あくまで人のデザインに有用であるためにはどうあるべきなのかという視点と、そのためにコンピュータをどのように利用すれば良いのかという方法論が重要となってくる。

1.2. 研究の目的

アルゴリズミックな手法によって人とコンピュータが協働するというデザインプロセスについてを、デザイン実例を通し、それらのデザインに実際に参加したという経験から考察する。

特に、コンピュータを用いてデザインに関する情報をアルゴリズミックに計算処理することと人間によるデザイン行為の関係、アルゴリズミックな手法によるデザインの効果、アルゴリズミックな手法によるデザインの方法論になどについての知見を得る。

1.3. 研究の意義

アルゴリズミックな手法によって、デザインにコンピュータが参加することの効果、優位性やそれを有効に働かせるための方法論を明らかにすることで、コンピュータを用いたデザインの可能性を広げることにつながると考える。

1.4. 研究の方法と構成

本研究では、実際のデザイン実例を通して、著者がそれらのデザインに実際に参加したという経験から、それらのデザインにおいてアルゴリズミックな手法が適用された経緯、適用された過程、それらをケーススタディとして分析することで、アルゴリズミックな手法によるデザインについての知見を得るという方法をとる。

これらのデザイン実例は、全てアルゴリズミックな手法が適用された事例である。個別のデザイン課題を扱ったものである一方、時系列的に連続したものであり、またその全てにおいてアルゴリズミックな手法の適用過程に著者が参加していたという共通点をもつ。

したがって本研究は、個別なデザイン事例へのアルゴリズミックな手法適用のケーススタディであると同時に、アルゴリズミックな手法のノウハウを複数のデザイン経験を通して、発展させていった過程の分析であるともいえる。これらを分析することで、アルゴリズミックな手法を個別のデザインに適応する上での知見のみならず、アルゴリズミックな手法によ

るデザインに通呈する考え方とそれらが発展した結果についての考察を行うことができる。

第 1 章において、研究の意義とその背景となる事柄についての説明と解釈を行う。そして 2 章において、各デザイン実例についての分析、考察をおこない、それらを通してアルゴリズミックな考え方によるデザインそのものについての考察を行う。3 章では、新たなデザイン事例に対して、それまでに得られた知見からアルゴリズミックな手法によるデザイン提案を行う。最後に 4 章で、本論文についての総括を述べる。

2. アルゴリズミックなデザインによるデザイン経験とその考察

2.1. アルゴリズミックなデザイン事例個々に対する考察

ここからは、著者自身が参加したアルゴリズミックな考え方に基づく実際のデザイン事例から、アルゴリズミックなデザインに必要な考え方、アルゴリズミックなデザインにおけるコンピュータと人の関わり方についての考察をまとめていく。

事例に対する各考察はまず、デザイン事例についての概要と著者が関わった内容を述べた後、そこからデザイン課題を抽出し、アルゴリズミックな手法を構築した過程についての考察を行う。次に、デザイン課題に適用した、アルゴリズミックな手法についてを、「デザイン・パラメータ」、「デザイン・アルゴリズム」、「デザイン・パターン」という 3 つの観点から解説し、それに考察を加える。これら 3 つの観点はデザイン事例 1 において設定したものであり、その詳細については、デザイン事例 1 の中で述べることとする。最後に、各事例の中で得られたアルゴリズミックな手法についての知見、次の事例に引き継がれた考え方などをまとめる。

2.1.1. デザイン事例 1 : Alternative Suburban - 卒業設計

「Alternative Suburban」は、著者が卒業設計、ならびに修士課程における研究課題の 1 つとして取り組んだ、戸建住宅街における住宅の配置デザインである。なおこの事例に関する研究は、ALGODE - International Symposium on Algorithmic Design for Architecture and Urban Design ならびに、第 34 回情報・システム・利用・技術シンポジウムにて、査読付論文として採用された。

このデザイン事例は、戸建住宅街における住宅建築群の配置デザインを対象としたものである。日本の郊外において、1960 〜 1980 年代にかけて活発に行われた新規住宅地開発の事例から、そうした住宅地がいくつかの問題を抱えており、それが住宅街の計画手法に起因するものと考えられることが分かった。こうした既存住宅街とその計画手法が抱える問題を解決するために、郊外 (Suburban) などにおいておこる大規模な新規住宅地開発において、既存の計画手法に対して、オルタナティブ (Alternative) となりえる新しい計画主手法を模索し、提案することを目的とした。

本事例において著者は、デザイン対象の設定から、デザイン提唱に呈する調査・分析、アルゴリズミックな手法の構築、デザインアウトプットの作成など、デザイン全般に携わり、それを主導した。研究、論文化に際しては、担当教授・池田靖史に協力をいただいている。

デザイン期間は約 1 年半ほどである。

2.1.1.1. デザイン課題の整理

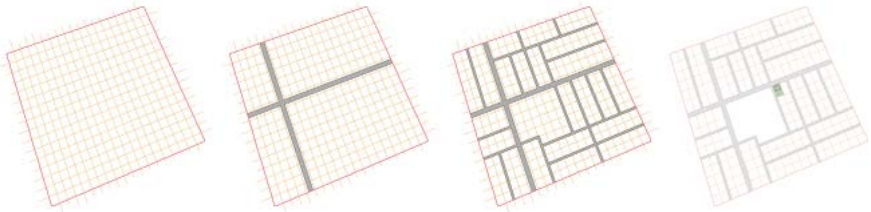
既存の戸建住宅街は、まず対象となる敷地を格子状に均等分割し、格子に沿ってヒエラルキーのある道路が配置され、最後に分割された個別の敷地において良い空間の実現が目指されるという計画手法がとられる。しかしこうした既存計画法では、住宅同士の空間的関係性が薄く、単調・均質な住宅街が出来上がってしまう。また、最初に全てを計画し一括的に開発するというトップダウン的な手法であるため、時間による住宅需要や環境的変化に対応できない、適応的でない住宅街が生成されてしまう。

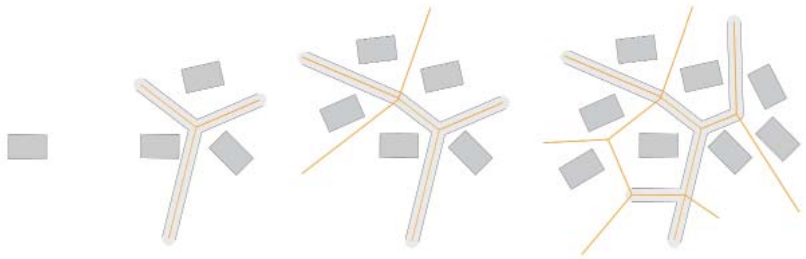
そのため、住宅街を住宅個々の空間的関係性によって構築するという、既存の手法とは逆方のボトムアップ的なアプローチを行うことで、既存住宅街と既存計画手法の問題点を回避できるのではないかと考えた。まず住宅街を構成する要素、住宅、住宅の外部空間、道路、公共空地などを、住宅とそれ例外の余白という要素に単純化した。そして、住宅同士がその余白を介して良い空間的関係性構成するように配置された後で、その余白を住宅の外部空間、道路、公共空地などに再設定するという手法を提案した。

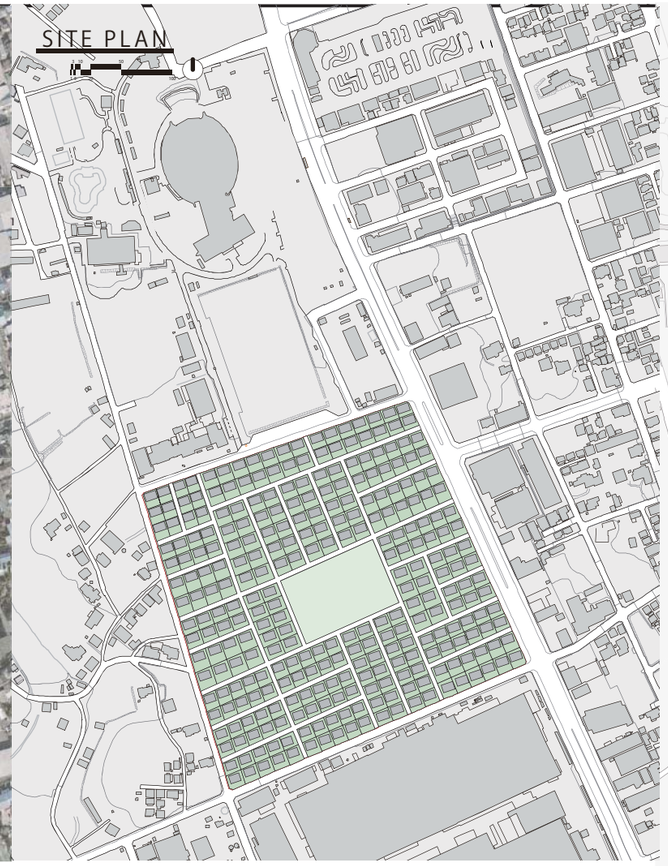
住宅の配置は段階的に行われ、道路や公共空地などもそれに応じて段階的に行われる。新しく住宅が配置される位置は、すでに配置されている住宅群との空間的関係性によって決定される。これは複雑系と呼ばれるシステムを参照したものである。。複雑系でみられる、部分を構成する条件によって系の挙動が決定される一方、それらが構築する全体は部分の条件とは異なる性質を獲得する場合がある。これは、住宅個々とそれが集まった住宅街全体という 2 つの条件を同時に考える必要がある住宅街に適した系であるといえる。また複雑系は、成長の全ての段階において、ある一定性能を満たしながら、変化に適応していけるという系でもあり、これも住宅街の問題を解決するために適している点である。

複雑系では、系を構成する大量の要素とそれらの相互関係を、随時把握しながら系の成長を実行する必要があり、また実行してみなければ、系の挙動がどうなるか分からないという問題もある。こうした問題を解決するために、コンピュータを用いて系の挙動をシミュレーションとすることで、系の挙動を把握し、それをコントロールしようと考えた。

本事例では、実在の敷地を参考にして設定された仮想敷地を、デザイン対象とした。







U-ichi Abe

DATE : 2010/01/27～2011/02/02
DRAWING : PRESENT & Layout

PROJECT : Alternative Suburban
SCALE : 1/3000 & 1/3000

PLACE : Endo, Fujisawa, Kanagawa, Japan
USE : Residential Area

2.1.1.2. デザイン・パラメータ

参考にした実在の敷地周辺の住宅規模を参照し、住宅の建築面積は 11m × 7m、住宅の高さは 10m とした。住宅は短手方向が南北方向となる状態を初期方向と、初期状態から回転させて配置してもよいが、その回転角の制限は東西どちらにも 45° までとした。。これは、住宅間の日照条件に極端なバラつきが生じないようにするためである。

住宅は初期方向における南面に決まった形状の空地を保障される。この空地は、任意の 7 種類の形状であり、住宅の回転にあわせて、常に南面から垂直方向に向かって配置される。南面にこうした空地を住宅南面に設けたのは、住宅南面の日照条件をよくするためである。7 種類ある空地の形状は、それぞれ形状に沿って他の住宅が配置されたとき、住宅同士が心地よい空間的距離感を保って配置されることを意図したものである。

住宅同士の間には一定以上近づいては行けない最小距離が存在する。また同様に、一定以上離れてはいけない最大距離も存在する。これは、住宅配置が過度にばらつかないようにするためと、隣棟感覚を保つためである。

住宅が新規建設されていくペースも設定されており、それは 1 年間に 7 棟というペースである。最後に各住宅は、40 年〜 60 年の間でランダムに選択された寿命が設定される。

こうしてみると、これらの「デザイン・パラメータ」には、住宅配置にとっての必要条件と十分条件、想定が可能な条件と本来は想定不可能であるはずの条件、数値的な状態を保障するための条件、感性的な状態を保障するための条件などが入り混じってしまっている点に大きな問題点がある。

2.1.1.3. デザイン・アルゴリズム

デザイン課題でも述べたように、本事例の「デザイン・アルゴリズム」では、住宅が 1 つづつ配置されていく。住宅が配置可能な位置は、対象敷地内で、かつ既に配置された住宅から最小距離以上、最大距離以内の範囲内限定される。住宅を配置した場合、既存住宅、既存住宅の外部空地と重なってしまうような位置も除外される。さらに、新規に配置される住宅は、その外部空地がいずれかの既存住宅の外部空地とつながるような位置である必要がある。新規住宅は、これらの配置条件を満たすために、制限角度以内であれば回転させて配置してもよい。これらのプロセスは、既存住宅と程良い空間的関係性を保てるような、新規住宅の位置と方向を模索するプロセスである。

以上のような条件の中で、住宅の配置位置はランダムに選択される。これはこの「デザインアルゴリズム」が、実際の住宅街を生成するルールとなった場合を想定し、予測のつかな

い人の挙動を再現しようと考えたからである。

アルゴリズムには、仮想の時間経過が実装されている。これは、「デザイン・アルゴリズム」が住宅街における時間経過に対して、適応的な対応ができるかを確認するためである。仮想の時間経過は、新規住宅配置のペースや寿命が尽きた住宅の消滅などに関係する。新規の住宅は 1 年間に 7 棟のペースで配置され、配置されてからの年数が寿命を上回った住宅は寿命が尽きたとして敷地から消去される。

この「デザイン・アルゴリズム」には終了条件が設定されておらず、「デザイン・アルゴリズム」の終了は人による操作によって行われる。本事例では 1 ～ 2 時間程度で住宅が敷地を充填し、2 ～ 3 時間程度で敷地を充填した住宅が新陳代謝により一新されるため、その時点で「デザイン・アルゴリズム」を終了させていた。

順次生成され、更新されていく全ての過程が「デザイン・パターン」として扱われる。

2.1.1.4. デザイン・パターン

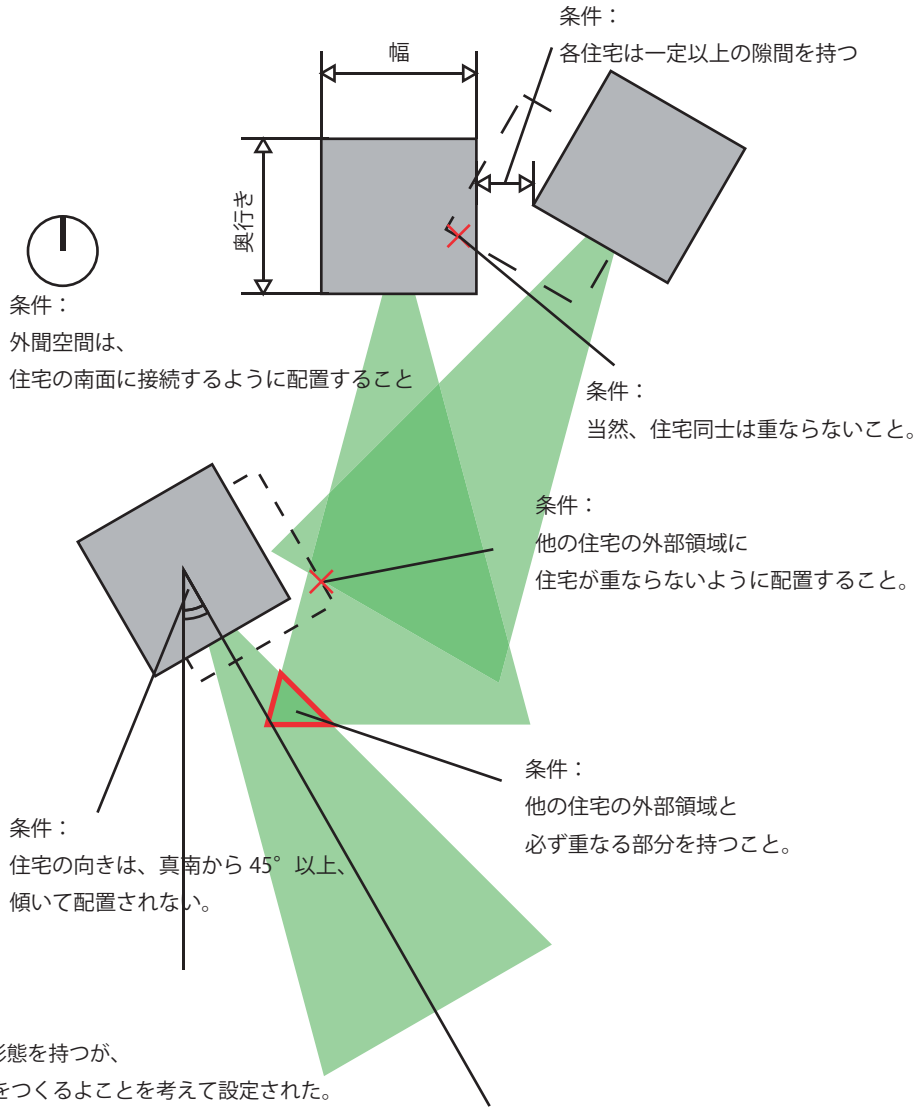
本事例では、複数生成された「デザイン・パターン」の中から、人が感性的な評価を経て 1 つの「デザイン・パターン」を選定し、最終的なデザインアウトプットとすることとした。しかし、生成された「デザイン・パターン」は評価困難であるか、あるいは良いと評価できないものであった。

パターンを生成したアルゴリズムは、デザイナーである著者の要求によってその意図を十分満たすように設計され、実行されたはずである。しかしデザイナーである著者は、その結果であるパターンに納得がいかないという状況が発生した。

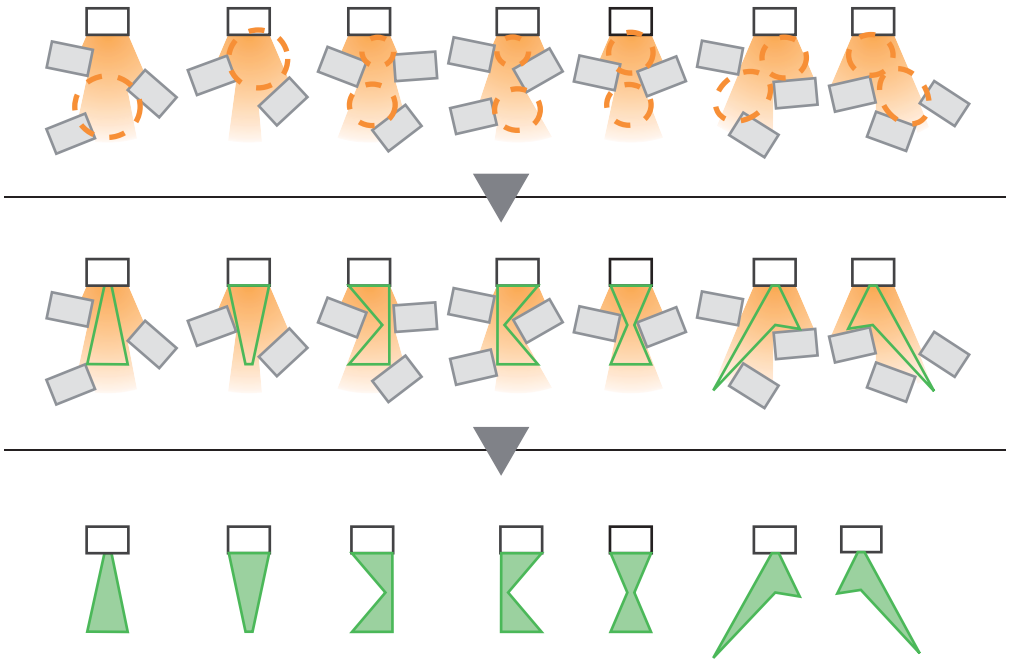
K･Terzidis は、コンピュータがアルゴリズミックな手法によって生成する結果は、アルゴリズムが設定された当初の意図と異なる場合があると述べている。そしてそれは、人にとって時に有効で啓発的であり、時に望ましくないものであるとも述べている。アルゴリズミックな手法を用いる際に、デザイナーはコンピュータに、こうした人にとって有効で啓発的な、意図しないパターンを提示してくれることを意識的、無意識的に期待しているといえる。

L. H. Sullivan は、*"形態は機能に従う。(form follows function.)"* という言葉を残しているが、著者はこの言葉のように、要求された機能を十分に満たすような優れた「デザイン・パターン」と「デザイン・アルゴリズム」からは、優れた「デザイン・パターン」が自動的に生成されると考えていた。しかし、人の感性に良いと評価される「デザイン・パターン」と機能的に生成された「デザイン・パターン」は必ずしも重なるわけでないことが分かった。本事例では、「デザイン・パラメータ」に人が感性的に決定した要素を入れたり、ということを実験的に行ってみたが、その結果は感性的な要因を記述することの難しさを確認しただけとなった。

「デザイン・パラメータ」として設定された条件



住宅に付属する外部空地は、7 つの形態を持つが、
それは住宅同士の間に、よい関係性をつくるよことを考えて設定された。



「デザイン･アルゴリズム」の解説

一番最初に配置される住宅の位置は、ランダムによって決定される。

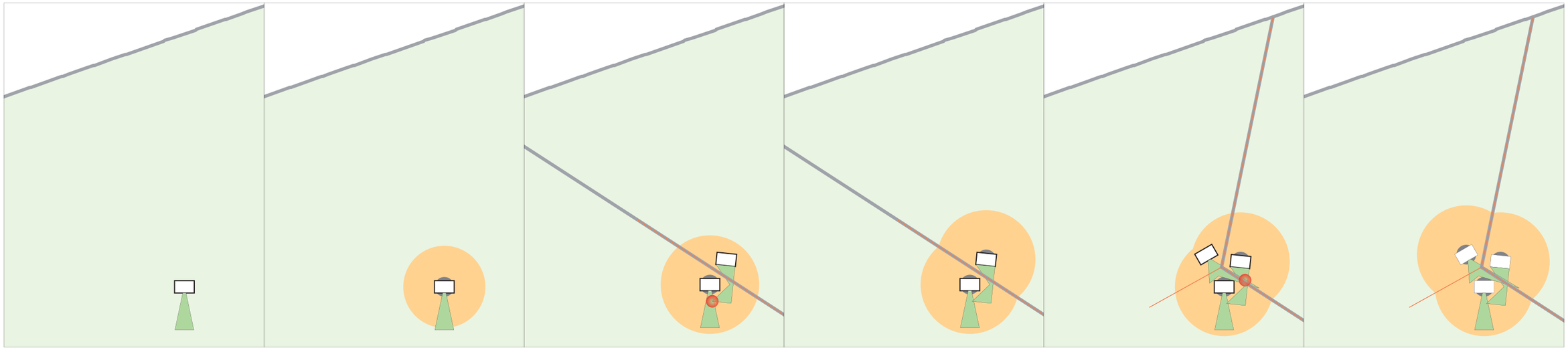
配置された住宅から、最大距離以内最小距離以内で、次に住宅の配置位置が模索される。

新しく配置される住宅は、真南から制限角度以内であれば、回転していてもよい。

また、住宅が南面にもつ外部空地の形状は、制限の中でランダムに選択される。

住宅の配置位置、回転角度、外部空間の形状は、ある制限によって選択される。

その制限とは、既存のいずれかの住宅の外部空地と重なっているも必要がある。



また住宅の外部敷地は、既存新規に関わらず、住宅と重なってはならない。

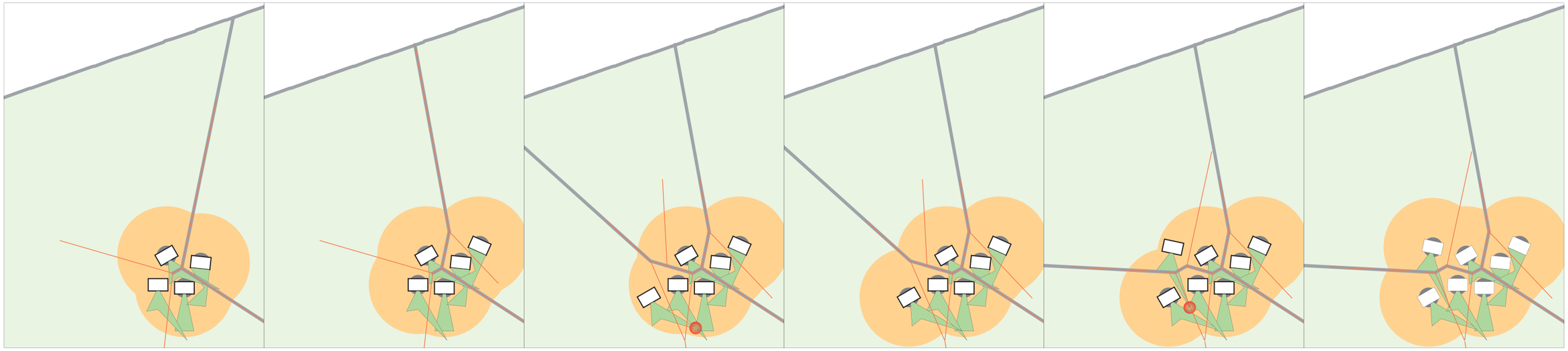
こうした制限の中から、住宅が 1 棟 1 棟配置されていく。

住宅が配置されると、住宅と住宅の間に、仮想の敷地分割線が引かれる。

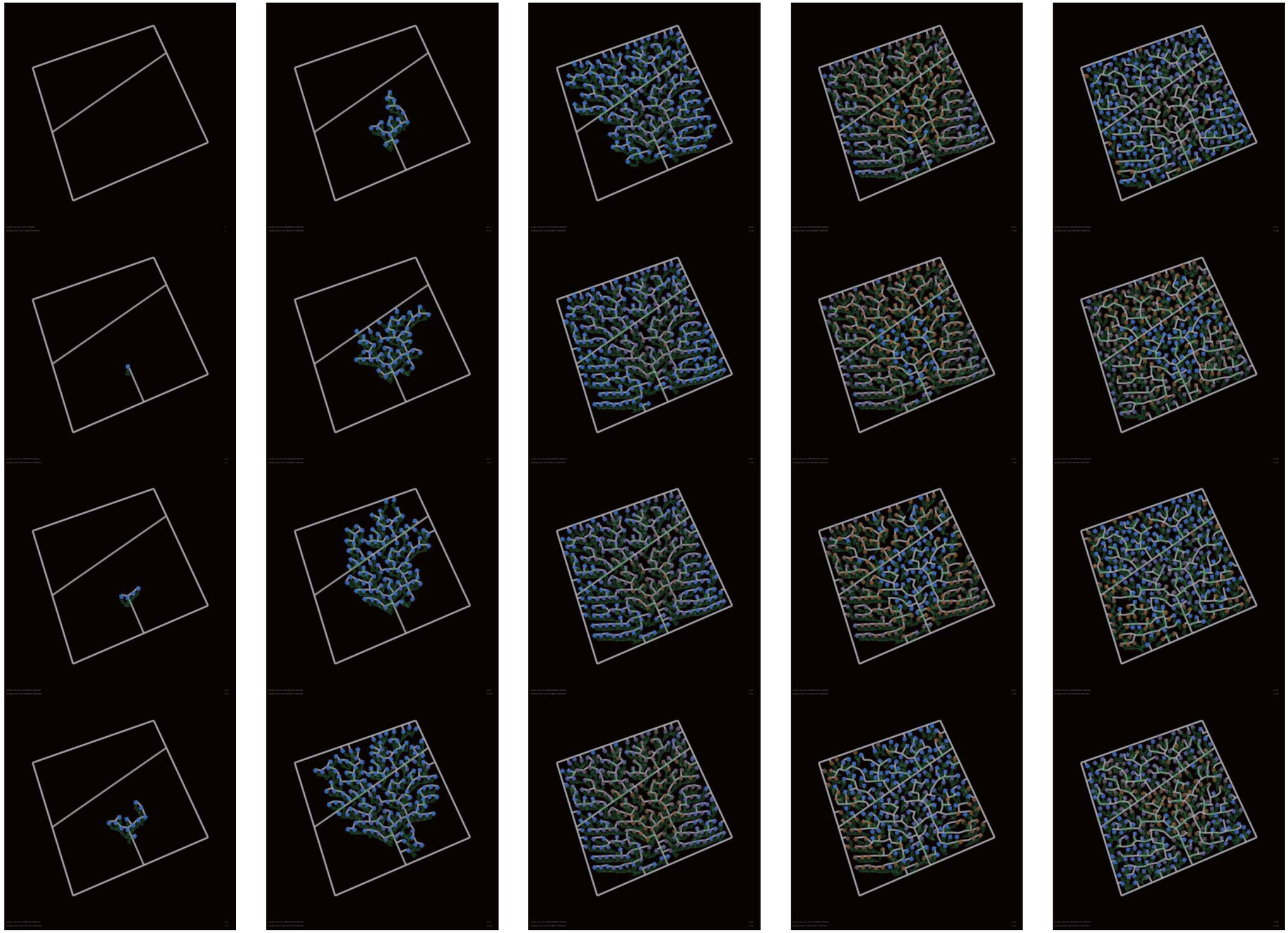
これは、住宅の生成が完了した段階で、実際の敷地境界線となる。

またこの線は、道路を配置する際の基準線にもなる。

道路は、住宅の生成に応じて、人の手によって引かれる。この作業を助けるのが、仮想の敷地分割線である。



「デザイン・パターン」の生成過程



2.1.1.5. 得られた知見

今回の事例では、アルゴリズミックな考え方によるデザインを考える上で指針となる、「デザイン・パラメータ」と「デザイン・アルゴリズム」から「デザイン・パターン」が生成される、というアルゴリズミックな考え方によるデザインの基本モデルを作成することができた。

「デザイン・パラメータ」とは、数学やコンピュータプログラムにおける変数に相当するものであり、デザインにおいては敷地条件や部材寸法などの初期条件となる要素や、「デザイン・アルゴリズム」において参照される、繰り返し回数や閾値などの要素である。

「デザイン・アルゴリズム」とは、数学やコンピュータプログラムにおける関数に相当する。「デザイン・パラメータ」を参照しながら、「デザイン・パターン」を生成していくプロセスのことであり、そのプロセスの手順を記述したものである。

そして「デザイン・パターン」は、「デザイン・パラメータ」と「デザイン・アルゴリズム」によって生成されるヴィジュアライズされた結果であり、デザインの解となるものである。

アルゴリズミックなデザイン手法において「デザイン・パターン」とは、結果的に生成されるものであり、直接的に操作されることはない。「デザイン・パターン」の操作は、「デザイン・パラメータ」を操作することにより間接的に行われる。

そのため、「デザイン・パターン」は意図通りに操作することが困難であり、反面「デザイン・パラメータ」は容易に操作可能であることが必要とされる。「デザイン・アルゴリズム」は、アルゴリズミックな手法において根幹となるものであり、もっとも重要な要素であると同時に、もっとも操作が難しい。アルゴリズミックな手法を検討する場合には、「デザイン・アルゴリズム」をどのように設定するかをもっとも慎重に考える必要がある。

また、「デザイン・パラメータ」と「デザイン・アルゴリズム」から、結果として生成されるということから「デザイン・パターン」は適応的なデザイン解であるといえる。なぜならば、「デザイン・アルゴリズム」を固定したままでも、「デザイン・パラメータ」を変更すれば、その「デザイン・パラメータ」にあわせた「デザイン・パターン」が再生成されるからである。

また、この「デザイン・パラメータ」の変更は、場合によっては「デザイン・パターン」の生成中でも可能である。それは、「デザイン・パターン」の生成に時間がかかる場合であり、今回住宅街という都市のデザインにアルゴリズミックなデザイン手法を用いたのも、都市が長い期間を経て生成されるものであり、かつその期間に起こる環境の変化に適応していく必要があるデザイン対象だからである。

一方で、人にとって優れた「デザイン・パターン」は要求された機能を満たすような「デザイン・パターン」と「デザイン・アルゴリズム」だけでは、生成できないことが分かった。それは、要求された機能を満たす「デザイン・パターン」と人が感性的によいと判断される「デザイン・パターン」は必ずしも重なるわけではなく、さらに人がアルゴリズミックな手法によってコンピュータを用いる場合、得られる「デザイン・パターン」に、「デザイン・パターン」と「デザイン・アルゴリズム」から導かれる以上の結果を期待しているからである。

ここから、コンピュータによるアルゴリズミックな手法について、2 つ課題が提示できる。1 つ目は、人の感性による判断とコンピュータによるアルゴリズミックな手法にどのように関係させていくかということである。本事例から、コンピュータによるアルゴリズミックなデザインプロセスにも、人の感性的な判断が必要不可欠になることが分かった。しかし、人の感性的な判断はコンピュータが苦手とするものであるため、コンピュータが「デザイン・パターン」を生成する過程に人の感性を導入する何らかの工夫が必要になってくる。

2 つ目は、人にとって有効で啓示的である「デザイン・パターン」をコンピュータに生成させるにはどうすればよいかということである。もちろん何の工夫がなくても、コンピュータによるアルゴリズムの実行が、アルゴリズム作成者の意図を超えることは起こりえる。しかしそれが、人にとって有効で啓示的であることは稀であり、多くの場合において、それは人にとって失敗やエラー、あるいは評価対象とならないというものになってしまう。

2 つ目の課題に対し、本事例から 1 つ述べることができる点がある。それは、コンピュータが生成した「デザイン・パターン」が人にとって有効で啓示的であるためには、コンピュータが「デザイン・パターン」を生成する時間が重要であるということだ。

「デザイン・パターン」が人にとって有用で、啓示的であるということは、「デザイン・パターン」が人にインタラクションを与えるようなものであるということである。アルゴリズミックな手法によるデザインにおいて人がインタラクションを得るというのは、「デザイン・パラメータ」を調整することで「デザイン・パターン」に変化が生まれ、そこから「デザイン・アルゴリズム」の妥当性や改良方法、「デザイン・パラメータ」と「デザイン・パターン」の関係性などを発見するといった過程のことである。

しかし、人がインタラクションを得るためには、対象となるものが人の感覚にとってちょうど良い速度で反応を返してくる必要がある。これは、人がある現象からインタラクションを得る場合には、その現象を注意深く観察し、類推と検証を何度も繰り返すという過程を経る必要があるからだ。しかし、本事例では「デザイン・パターン」の生成に時間がかかるため、そうした繰り返しの過程を十分には行えなかった。

これは、アルゴリズミックな手法においてコンピュータが行う処理にも、潜在的な上限が存在するということを示している。コンピュータが「デザイン・パラメータ」を生成する速度は、「デザイン・パラメータ」による制限条件の厳しさと、「デザイン・アルゴリズム」の複雑さ、そしてデザイン要素の数量に起因する。コンピュータは本来、複雑な膨大な計算でも、長大なアルゴリズムでも、根を上げることなく実行することが可能である。しかし、アルゴリズミックな手法によるデザインにおいて、「デザイン・パターン」の生成を人がインタラクションを得やすい速度で行うという条件を守るためには、そうした処理に制限を設ける必要がある。

2.1.2. デザイン事例 2 : Porous Torus

「Porous Torus」は、屋内展示会場における展示空間のための木造仮設建築物のデザインプロジェクトである。2010 年 11 月 22 日～ 23 日に六本木ヒルズ森タワー・アカデミーヒルズ 40 において行われた、慶應義塾大学湘南藤沢キャンパスの研究発表展示会 SFC Open Research Forum 2010 において、池田靖史研究室の研究成果、ならびに展示ブースとして、デジタルファブリケーションによる仮設建築物をデザインした。

本事例は、池田靖史研究室が研究会として取り組んだプロジェクトである。デザイン対象の設定、デザイン対象に対する調査・研究、アルゴリズムの検討については、著者は研究会の一員として他の研究会メンバーと協働で行った。アルゴリズムのコンピュータへの実装については、著者のみが担当した。

研究、論文化に際しては、担当教授・池田靖史に協力をいただいている。

デザイン期間は約 2 ヶ月であった。

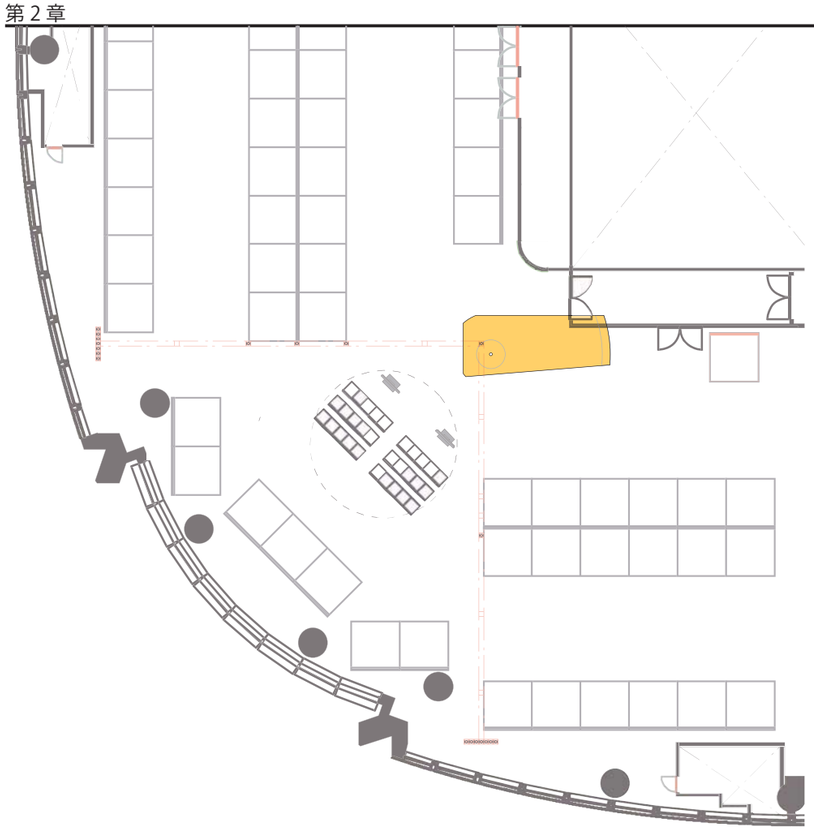
2.1.2.1. デザイン課題の整理

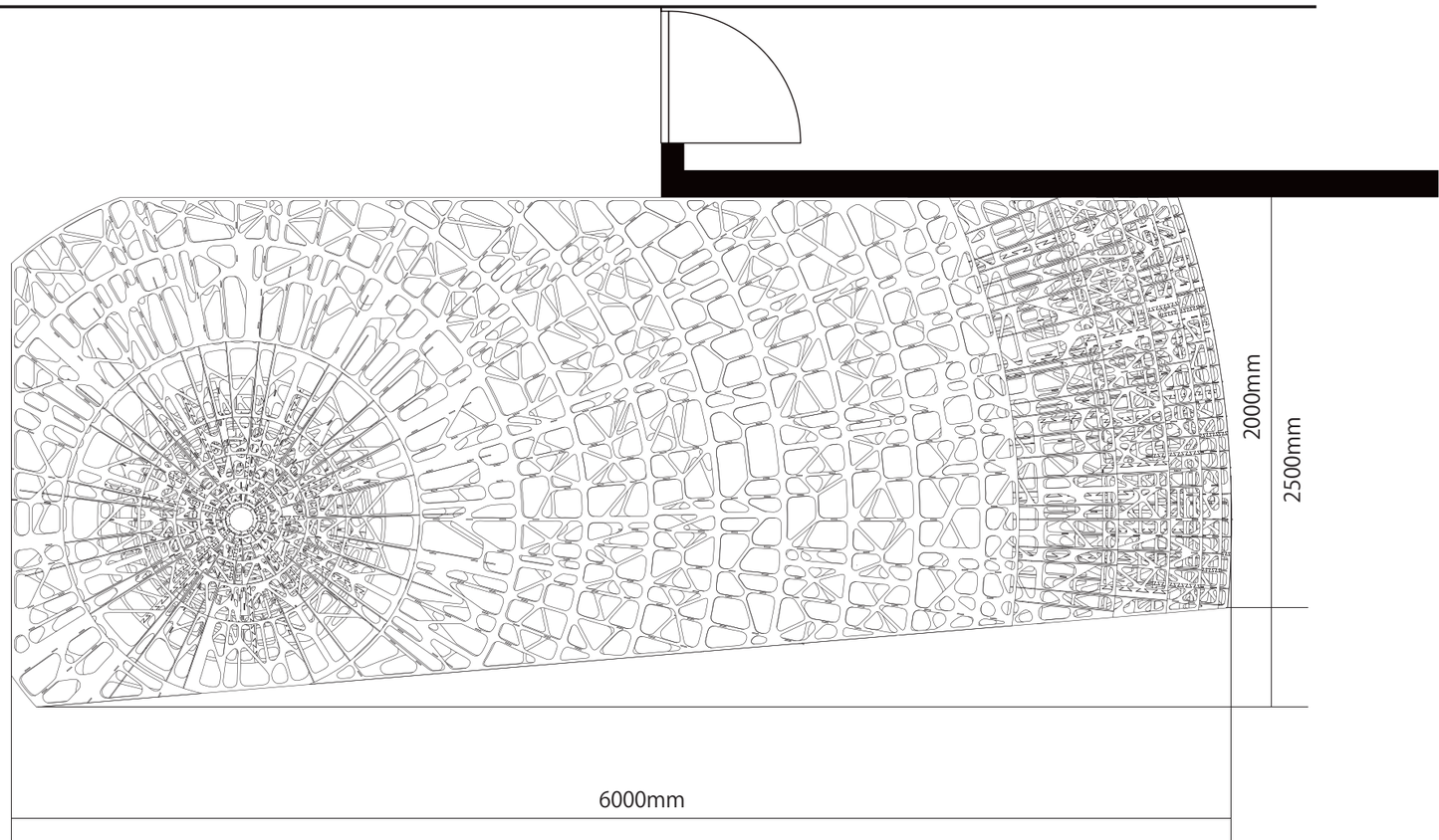
対象の仮設建築物は、与えられた展示スペースの空間を最大限利用するためと、生産、運搬、施工の制限から、小規模のパーツをアッセンブルして 6m × 2m × 1.8m の展示空間を覆うような構造物と設定された。また、一時的な展示という用途から、施工、解体に時間のかからないジョイントシステムを採用する必要があった。さらに、建物内における展示と木という材質から、防火対策を行う必要があり、スプリンクラーの放水が透過するように多孔質な物にする必要があると同時に、実物の建築であるという条件から、構造的な条件も同時に満たす必要があった。

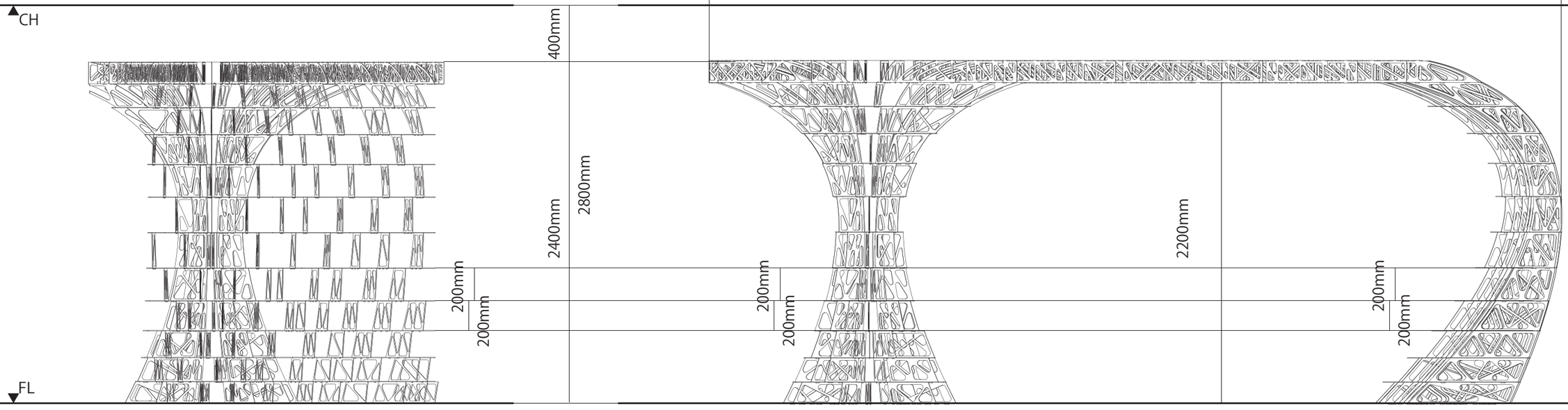
こうしたデザイン条件から、対象の仮設建築は大量の部材によって構成され、その部材は相互に関係しあう位置関係をバランスした状態で組み合わさっている必要があった。このデザイン課題を解くために、形状全体を定義できるグリッドを設定し、1 グリッドを 1 セルとするセル空間において、セルラ・オートマトンのように生成的に部材位置の関係を決定していくというアルゴリズミックな手法が設定された。

全ての部材位置の関係を把握するためには、仮設建築全体を定義できるジオメトリが必要となり、それは同心円と放射線によるグリッドであるとされた。同心円と放射線のグリッドにおいて、放射線方向に並ぶセルは部材同士の上下関係を規定し、同心円方向に並ぶセルは同じ段における水平方向の関係性を規定する。このグリッドによるセル空間は、部材位置の関係を把握するためには十分であるが、直行グリッドと違い、セル 1 単位と実空間で対応する寸法がセルの位置によって異ってしまう。そのため、セルの位置によって、部材の位置関係を決定する条件を調整するというプロセスも必要となった。

こうした大量の要素が相互関係を持つような条件において全体の状態を決定することは、人によるデザインでは困難である。さらに、セルの位置によって部材の位置関係を決定する条件をも調整しなければならないとなっては、このデザイン課題を人の手によって解決は不可能である。そのため、コンピュータによってこうしたアルゴリズミックな手法を実行することで、部材配置を決定することとなった。







Yasushi Ikeda Lab. / Keio Univ. SFC	DATE : 2010/11/22 DRAWING : ELEVATION & Layout	PROJECT : Porous Torus SCALE : 1/60 (only Layout 1/500)	PLACE : Academy Hills 40, Roppongi Hills Mori Tower, Tokyo, Japan USE : Temporary Architecture for Exhibition Site

2.1.2.2. デザイン・パラメータ

放射線方向のセル数は、部材の段数に対応し 32 マスと設定された。同心円方向のセル数は、いくつかの試行を経て 1 周 360° を 0.5° づつ 720 分割した 720 マスと設定された。また以降説明の都合上、放射線方向のセルの並びを段。同心円方向のセルの並びを列、セル空間において中心に向かう方向を上、外周に向かう方向を下、時計回り方向を左、反時計回り方向を右と呼ぶ。つまり、このセル空間は 32 段 720 列であり、原点となる 0 段 0 列目のセルは右下に位置する。

このセル空間において、段は水平部材、列は水平部材の分割位置と水平部材に縦部材が差し込まれる位置の候補位置をあらわす。したがってこのセル空間では、どのセルが分割位置、あるいは差込位置であればよいかが検討される。縦部材の差込位置と水平部材の分割位置は、隣の要素と近づいてもよい最小間隔と離れてはいけない最大間隔が設定されている。縦部材の差込位置の最小間隔距離は 50mm、最大間隔距離は 300mm である。水平部材の分割位置の最小間隔距離は 50mm、最大間隔距離は 750mm である。

すでに述べたように、同心円と放射線によるセル空間では、1 セルが対応する寸法距離は中心に近づくごとに小さくなる。そのため、実際の形状でその段が対応する位置の周長と、上記の最小・最大間隔距離をもとに、各段ごとの最小・最大間隔セル数が設定される。たとえば周長が 7200mm である場合、1 セルが対応する距離は 100mm となるため、縦部材差込位置の最小間隔セル数は 1 セルとなり、最大間隔セル数は 3 セルとなる。

2.1.2.3. デザイン・アルゴリズム

前述したとおり、最終的な成果として期待されている各部材の適切な配置は、分割位置と差込位置によって決定する。その検討は 32 段 720 列のセル空間において行われ、どのセルが分割位置、あるいは縦部材差込位置として設定されれば良いかという問題として計算される。また縦部材は上下の水平部材に差込位置を必要とするため、ある段でのセルの状態は 1 段下の差込位置セルに影響を受ける。

セルの状態計算は段毎に行われ、分割位置となるセルと差込位置となるセルが端から順に設定されていく。新しい分割位置セル、あるいは差込位置セルは、同段で先に設定された分割位置セル、差込位置セル、そして 1 段下の分割位置セルから、それぞれの最小間隔セル数と最大間隔セル数を守るように指定されていく。

セルの状態を下から、あるいは端から決定していった場合、途中でアルゴリズムが手詰まりを起こす場合がある。最大間隔セル数以内の候補セル全てが、他の設定済みセルの最小間

隔セル数以内にあるような場合である。こうした状況が発生した場合は、その段の状態を初期化し、再度セルの状態を計算しなおす。同じ段において、特定の回数初期化と再配置が繰り返されてもその段の状態が決定しなかった場合は、1 つ下の段までを初期化しその 1 つ下の段から配置計算をやり直す。特定回数 1 段下に戻ることが繰り返された場合には、2 つ下の段までを初期化し、その 2 つ下の段から配置計算をやり直す。

2.1.2.4. デザイン・パターン

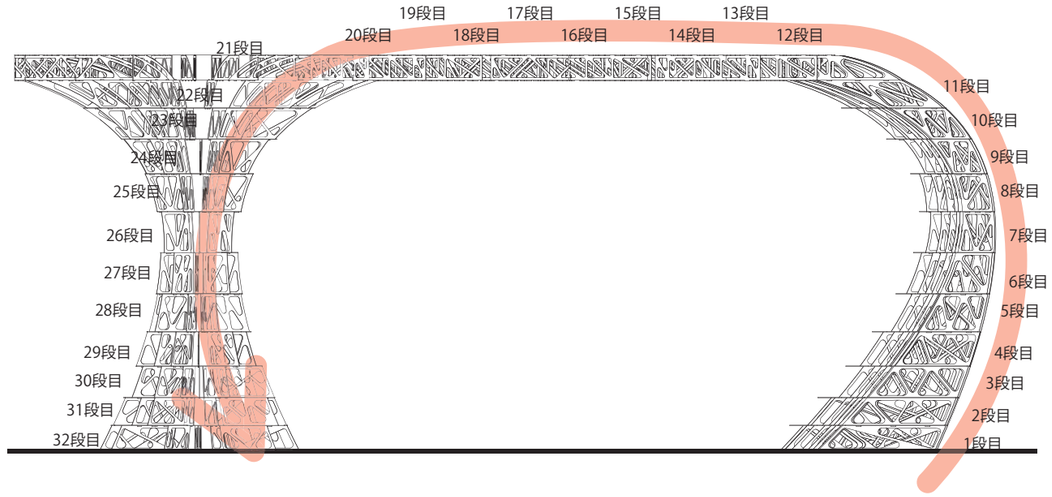
本事例では、「デザイン・パターン」そのものだけでなく、「デザイン・パターン」の生成されるプロセスが重要となった。それは、「デザイン・パラメータ」として設定された制限条件によって、「デザイン・アルゴリズム」が手詰まりをおこし、「デザイン・パターン」が生成されないという可能性が存在したからだ。そのため、設定された「デザイン・パラメータ」が「デザイン・パターン」を生成できる適切なものかどうかを判断するために、「デザイン・パターン」の生成プロセスを観察する必要があった。実際にコンピュータが何度やり直しても「デザイン・パラメータ」を生成できなかった場合、「デザイン・パラメータ」による制限を調整することで、適切な「デザイン・パラメータ」の模索を行った。つまり、「デザイン・パターン」が最終的なデザイン解としてだけではなく、「デザイン・パラメータ」が適切かどうかの判断材料としても位置づけられていたということだ。

さらにこのことから、今回の「デザイン・パターン」は、生成がされたとということ自体が、明確で肯定的な判断基準になりえたということが分かる。「デザイン・パラメータ」と「デザイン・アルゴリズム」は、デザイン課題から導かれた、具体的な問題解決の手段であったため、「デザイン・パラメータ」と「デザイン・アルゴリズム」に従って、「デザイン・パターン」が生成されたということ自体に価値があったのだ。

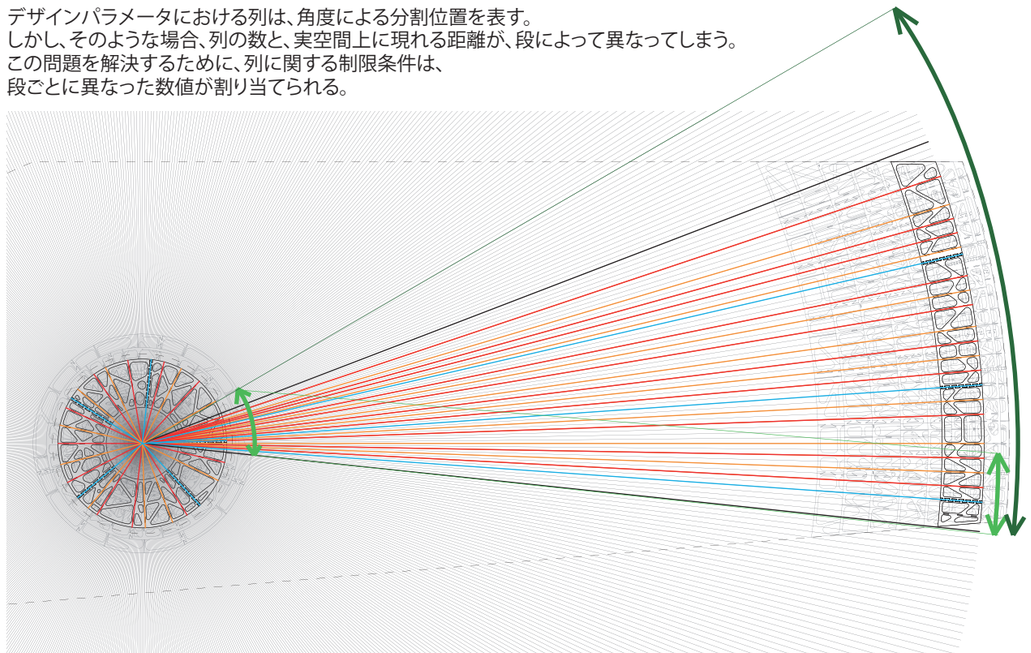
デザインパターンの生成過程で判断を下す場合があった。

「デザイン・パラメータ」と「デザイン・アルゴリズム」の前提条件の解説

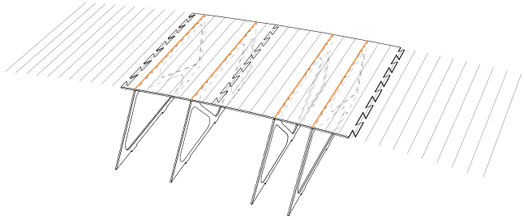
デザインパラメータにおける段は、以上のように数える。
天井部分の平らな場所においても、段と段に働く相互関係は変わらないため、
このような1続きの段として扱える。



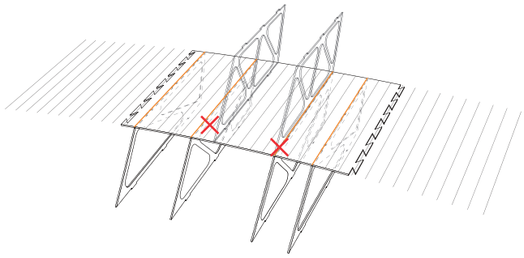
デザインパラメータにおける列は、角度による分割位置を表す。
しかし、そのような場合、列の数と、実空間上に現れる距離が、段によって異なってしまう。
この問題を解決するために、列に関する制限条件は、
段ごとに異なった数値が割り当てられる。



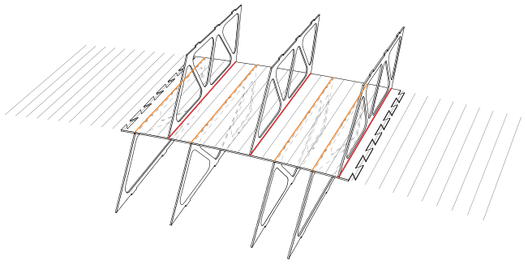
水平部材の分割位置と、縦部材の差込位置には、
右のような制限がある。
前提として、それらはセル空間において、
等しくセルが保持する値として解釈される。



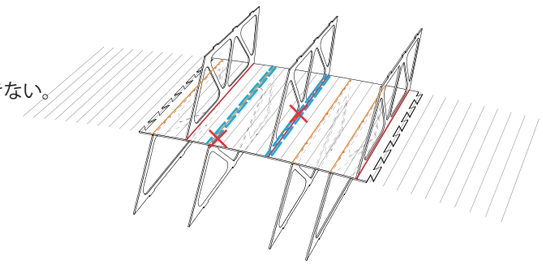
下の段の縦部材差込位置がある場所には、
縦部材差込位置を配置できない。
また、まったく同じ位置出なかったとしても、
その段で設定されている最小セル数以内であれば、
同じように配置はできない。



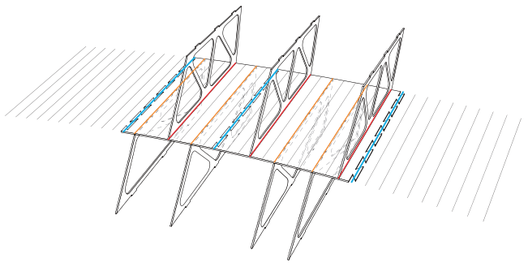
こうした条件を避けるように、
縦部材の差込位置は決定される。



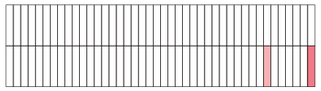
水平部材の分割位置も、
下の段の縦部材差込位置がある場所におくことができない。
また、既に配置されている、
同じ段の縦部材差込位置とも重なることができない。
これは縦部材差込位置もどうようで、
同じ段で、既に配置されている水平部材分割位置に、
縦部材差込位置を置くことはできない。



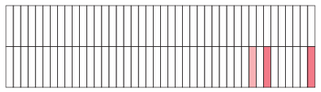
こうした2つの要素、
縦部材差込位置と水平部材分割位置の関係によって、
その段の部材位置が全て決定される。



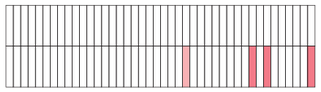
模式的なモデルによる、「デザイン・パターン」生成プロセスの説明



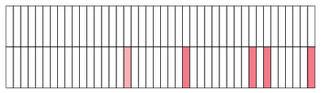
まずは、最初に位置に縦部材部材が配置される。セルが赤くなっているのがその場所である。



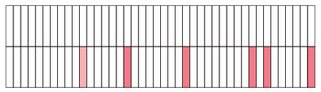
配置され部材から最小間隔セル数以上、最大間隔セル数と以内のセルを次の部材配置位置を選ぶ。



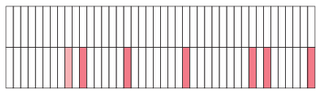
そのセルに部材を配置する。



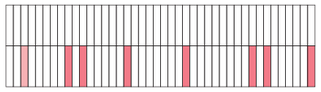
最大間隔セル数と最小間隔セル数の範囲のであっても、その選択に自由度がある場合がある。



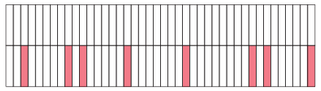
こういった場合にどれを選択するかは、ランダムによって決定する。



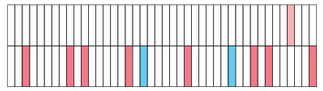
最初の段では制約がゆるいため比較的自由に部材の配置位置を決定することができる。



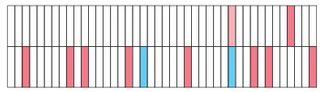
そのため、ほとんどの選択がランダムにしたがっている。



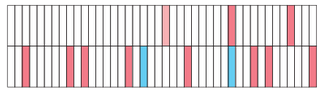
その段の縦部材の位置決定が終了する。



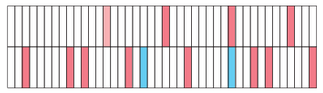
1段上の段に進み部材の配置が行われる。



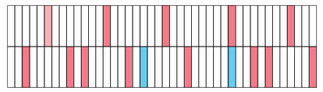
今回は、1段下の段で、縦部材が配置されているセルには、部材が配置できない。



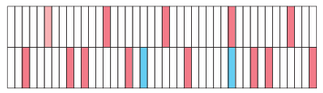
そのため、配置可能位置が大きく制限される。



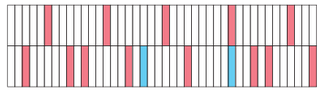
さらに、最大間隔セル数と最小間隔セル数が、下の段と異なる場合もある。

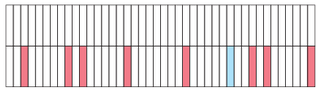


これも、部材の配置位置を大きく制限する。

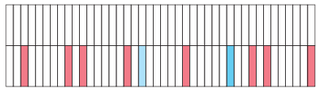


次の部材の候補位置が見つからなくなると、その段を白紙に戻し1から配置をやり直す。

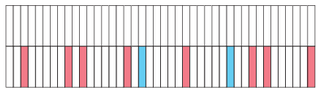




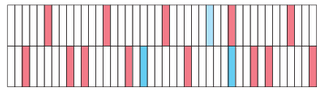
次に、水平分割線の位置が決定されていく。



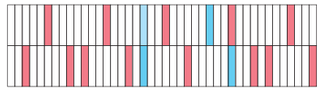
手順は縦部材の際と同じであるが、既に同じ段の縦部が既に配置されているため、それを避けなければならない。

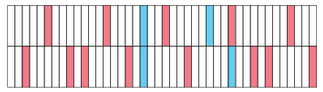


水平部材の配置が終了する。



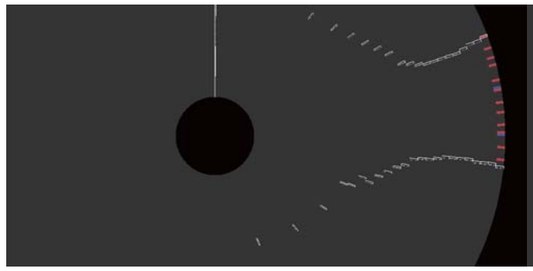
もし同じ段で、何度も配置に失敗する場合は、2段前までを白紙に戻し、配置をやり直す。



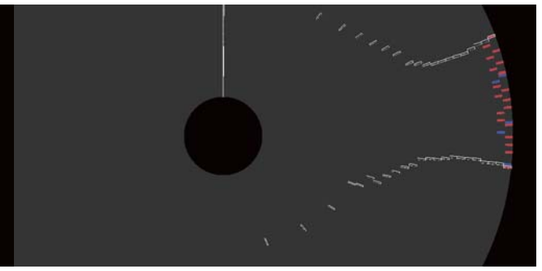


コンピュータによる「デザイン・パターン」生成プロセス

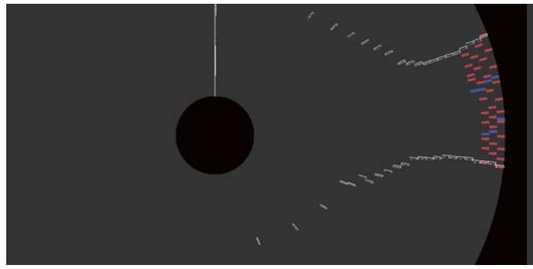
1st step



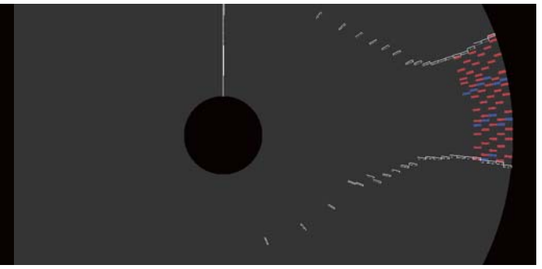
50th step



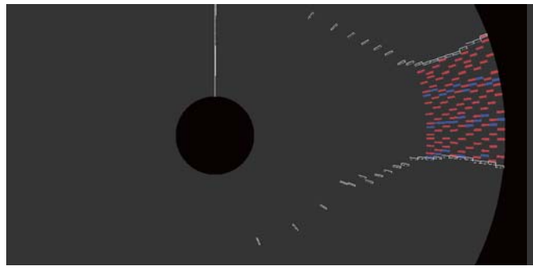
100th step



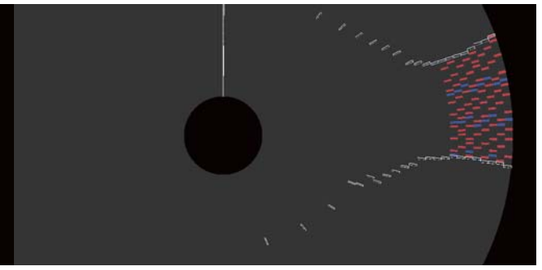
150thstep



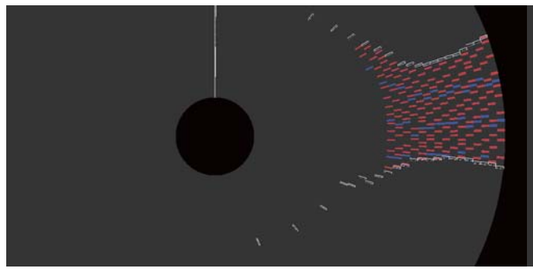
200st step



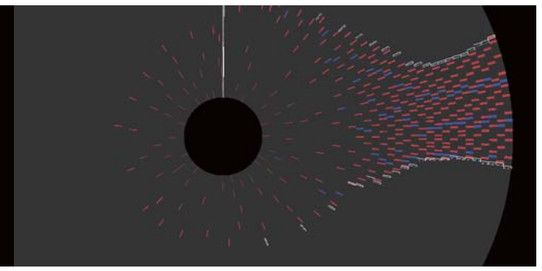
250thst step



300st step



358th step (completed)



「Porous Torus」の施工の様子と、完成写真

2.1.2.5. 得られた知見

デザイン事例 2 では、アルゴリズミックな手法によるデザインが良いデザイン解を得ることのできるデザイン手法であること、また既存のデザイン手法よりも優れた点があることが確認できた。これはデザイン解が満足のいくものではなかったデザイン事例 1 と、大きな違う点である。

こうした違いの理由は、対象となったデザイン課題の性質の違いと、それによる「デザイン・パターン」に求められていたことの差にあると考えられる。今回のデザイン事例には、明確でロジカルに記述可能なデザイン課題が存在した。それは、今回のデザイン対象は実現を前提ものであり、思考実験としての側面が強かったデザイン事例 1 よりもデザイン課題が具体的なものであったからだ。そして「デザイン・パラメータ」と「デザイン・アルゴリズム」は、その具体的なデザイン課題に対して設定された。今回の事例では、こうして設定された「デザイン・パラメータ」と「デザイン・アルゴリズム」から「デザイン・パターン」が生成されること自体に重要な価値があった。

人の感性による要求はそれに比べれば優先順位が低く、また求められていたことも、部材の配置が均質ではなく、だが偏ってもいないという曖昧に定義された状態を達成するということであった。こうした状態を「非均質・非偏在」の状態と呼ぶ。「非均質・非偏在」の状態を実現することは、デザインにおいてその目的となる場合も多い。しかし「非均質・非偏在」の状態とは、人の感性によって曖昧に定義されものであり、それゆえ広い許容範囲を持つ。こうした広い許容簡易を持つ状態をデザインの目的としたことが、本事例がよいデザイン解を得ることができた理由の 1 つだと考える。

またアルゴリズミックな手法によるデザインでは、複雑なデザイン課題を扱うことが多く、生成された「デザイン・パターン」も複雑なものとなる場合が多い。複雑な「デザイン・パターン」とは、それぞれに様々な状態を持つ要素が、大量に集まってできている「デザイン・パターン」のことあるが、人の処理能力には限界があるため、こうした多くの状態を持つ大量の要素を適切に評価できない。また複雑な「デザイン・パターン」には、無数の微妙に異なるバリエーションが存在するため、それら全てに感性的に優劣がつけられるような差異が、そもそも存在するのかどうかも不明である。

いくら「非均質・非偏在」という目的の許容範囲の大きいとは言え、「デザイン・パターン」が「非均質・非偏在」であるかどうかが判断できなければ意味はない。またデザイン課題によっては「非均質・非偏在」な「デザイン・パターン」を実現することだけではなく、実現された「デザイン・パターン」がさらに人の感性的にもよいと判断されることを求める場合もありえる。デザイン事例 1 には、正にそういったデザイン課題が設定されていた。

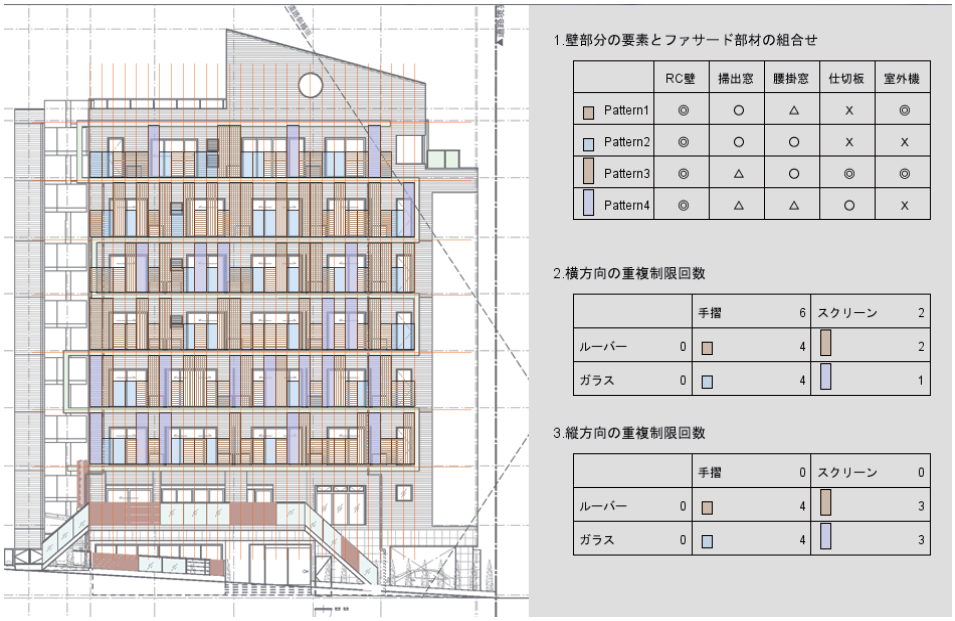
デザイン事例 2 において生成された「デザイン・パターン」は、セル空間によって表現された。生成された「デザイン・パターン」は確かに複雑ではあったが、そこにはあらかじめ配置が均質であるセル空間という制約がかけられていた。このことが、生成された「デザイン・パター

ン」の複雑さを人が評価可能な程度に抑え、人はそのパターンが「非均質・非偏在」な状態であることを判断することができた。

また、複雑な「デザイン・パターン」を人が評価する際には、人に明確な評価基準と評価対象を提示する必要がある。今回の事例においても、生成された「デザイン・パターン」が、「非均質・非偏在」な状態であるかを判断するために、似たようなパターンが規則的に繰り返して出現していないかという評価基準が設けられた。

今回の事例では、1 つの「デザイン・パターン」が生成されるのに 20 分〜 1 時間程度の時間がかかった。これはデザイン事例 1 において挙げられた、人がインタラクションを得られる時間からすると若干長いように思う。しかし今回の事例では、「デザイン・パターン」の生成から十分なインタラクションを得ることができた。これはデザイン課題の要求から、「デザイン・パターン」が生成できること自体に価値があり、「デザイン・パターン」が生成されていく途中のプロセスからもインタラクションを得ることができたからだ。実際に「デザイン・パターン」の生成が終了することを待たずに、「デザイン・パラメータ」や「デザイン・アルゴリズム」が適切でないことに気づき、それを修正するようなことがあった。

このように、人が「デザイン・パターン」からインタラクションを得る際に重要な「デザイン・パターン」生成の時間とは、「デザイン・パターン」の生成が終了するまでの時間ではなく、「デザイン・パターン」の生成が人が評価可能な段階にいたるまでの時間速度なのである。もちろん人にとって評価対象になるのが、生成が完了した状態の「デザイン・パターン」である場合は、「デザイン・パターン」の生成が終了する間での時間が問題になる。しかし、デザイン・パターン」の生成過程そのものが、人にとって評価の対象となる場合、「デザイン・パターン」の生成が終了する間での時間はそれほど問題ではなくなる。



2.1.3. デザイン事例 3 : ドミエメロード / ファサードデザイン

「ドミエメロード / ファサードデザイン」は、「ドミエメロード」という 1 ～ 2 階が商業、3 ～ 8 階が住居という複合用途建築において、そのファサードデザインを行うというプロジェクトである。4 種類のファサードパネルを組み合わせることで「ドミエメロード」のファサードをデザインすると同時に、デザイナーからはそのデザイン検討に施主が参加できるようにしてもらいたいという要求があった。、

本事例は、連健夫建築研究室、IKDS と協働して行われたプロジェクトである。連健夫建築研究室が基本設計を行った「ドミエメロード」において、連健夫建築研究室からそのファサードのデザインと施主のデザイン参加の実現という課題が提示され、それを実現するアルゴリズムの検討を、連健夫建築研究室、IKDS、著者の 3 者でおこなった。その後、著者が担当となって、コンピュータへの実装を行い、施主、連健夫研究室、IKDS、著者の 4 者で、デザイン決定のための WS を行った。

デザイン期間は、全体で 2 ヶ月ほどであった。

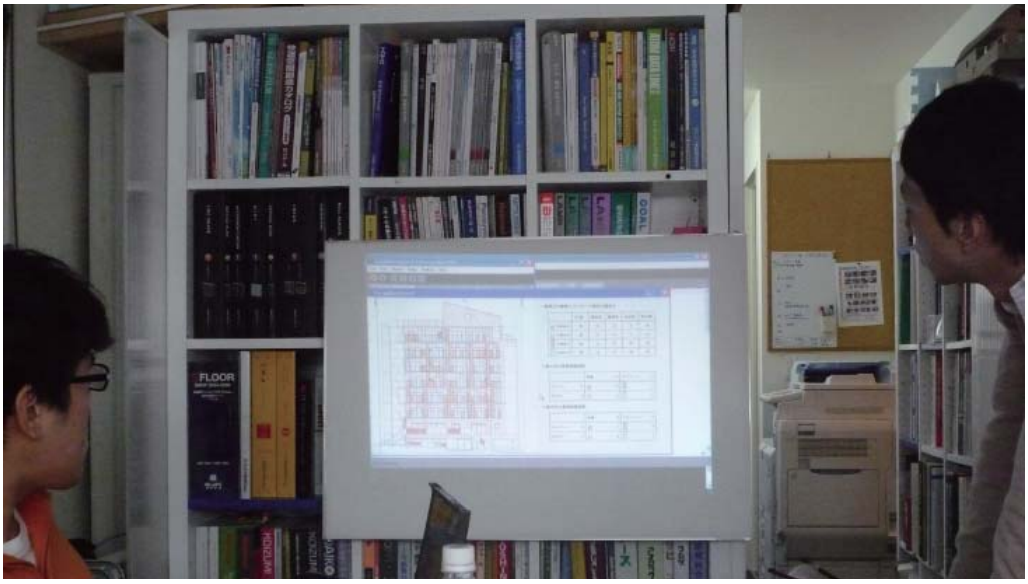
2.1.3.1. デザイン課題の整理

デザイン対象が設定された段階で、既にファサードに使用されるパネルの種類は決定されていた。これらのパネルは、ベランダの外側にカーテンウォールのように設置されるものであった。ベランダの内側には、内外を隔てる建築本来の壁面があり、そこには窓や室外機、ベランダの仕切板などが、すでに配置されていた。こうした、パネルの種類、ベランダ内側の状態を極力変更せずに、デザイン課題に取り組むことが必要とされた。

またこのファサードパネルは、マンションのファサードを構成するだけでなく、ベランダの手摺や、室外機や仕切板を隠す目隠しとしての機能も与えられていた。他に消防法の規定から、特定のパネルには横方向に一定数までしか連続して配置できないといった制限もある。デザイナーの要求は、既存の建築壁面の状態にあわせ、決められたパネルを使用して、これらの機能的な条件を満たしつつ、「非均質、非偏在」の状態を施主がインタラクションを得られる形で生成することであった。

本事例最大の特徴は、デザイナーでもアルゴリズム設計者でもない、施主という新しい主体をデザインに参加させることを目的の 1 つにおいていたことだ。そのため、アルゴリズミックな手法は、どれだけ明快なインタラクションを与えられるかという視点で検討された。事例 1、2 でも考察したように、人がインタラクションを得るためには「デザイン・パターン」の生成速度で生重要になるため、「デザイン・パターン」の生成はコンピュータが担当した。

こうした諸々のデザイン課題から、パネル配置位置を 1 セルとするセル空間を用意し、そのセル空間においてファーサードパネルの配置を検討することとした。マップレイヤーというファサード内側の壁面状態を記録するセル空間と、実際にファサードの配置検討をするソリューションレイヤーというセル空間の 2 つを用意し、ソリューションレイヤーにおいて、同種類のパネルが縦、横に何回まで連続できるかという条件と、マップレイヤーの情報を参照しながら「デザイン・パターン」を生成させた。マップレイヤーとソリューションレイヤーの関係性、縦横の連続回数制限という「デザイン・パラメータ」を操作することで、「デザイン・パターン」制御することとした。



MURAJI TAKEO ARCHITECTURAL LABLATORY & IKDS

DATE : 2011/04/22
DRAWING : ELEVATION

PROJECT : Domi Emeroude
SCALE : 1/150

PLACE : Akasaka, Tokyo, Japan
USE : Commercial & Residence

2.1.3.2. デザイン・パラメータ

セル空間は、直行グリッドによる 6 段 28 列のセル空間から、最上段最後の 2 セルが欠けた少し変則的な形である。つまり、1 ～ 5 段目は 1 ～ 28 列まであるが 6 段目は 1 ～ 26 列までしかないということである。

まずは、ファサードパネルが配置される内側壁面の状態を写し取ったマップレイヤーが設定された。マップレイヤーにおいて各セルは、0. RC 壁　1. 掃出窓　2. 腰掛窓　3. 仕切板　4. 室外機　という 5 つの属性の中からいくつかの値を持つ。これは、セルの位置に掃出窓と室外機が両方存在するような場合があったからである。

ソリューションレイヤーにおいて配置されるファサードパネルは、1. 木目調アルミルーバー手摺 (以降、ルーバー手摺)　2. ガラス手摺　3. 木目調アルミルーバースクリーン (以降、ルーバースクリーン)　4. 摺りガラススクリーン (以降、ガラススクリーン)　の 4 種類がある。全てのパネルは 600mm の横幅を持ち、手摺とは高さ 1200mm のパネルであり、ベランダの床面から天井面までを覆うパネルである。

どのマップ要素の前にどのパネルの候補がありえるかという条件は、マップ要素それぞれに対して 4 段階で設定される。その 4 段階とは、配置すべき (◎)、配置しても良い (○)、配置しないほうが良い (△)、配置してはいけない (×) である。例えば、3. 仕切板　の前は、手摺が来てしまうと仕切板が隠せないため、1. ルーバー手摺　と 2. ガラス手摺　は (×)、3. ルーバースクリーン　と 4. ガラススクリーン　は、どちらが選ばれてもよいため、両方共 (◎) となる。また、マップ要素が重なり合っている場合は、この条件も重ね合わせられる。例えば、3. 仕切板　と 4. 室外機　が重なり合っていおり、さらに 4. 室外機　の条件が 1. ルーバー手摺　3. ルーバースクリーン　は (◎)、2. ガラス手摺　4. ガラススクリーン　は (×) であった場合、このマップセルの前に配置できるパネルは 3. ルーバースクリーンのみとなる。

各パネルには、ソリューションレイヤーにおける配置の際、何回まで同じパネルが連続できるかという上限が決まっている。これは、縦、横両方向についてそれぞれ設定されている。また、手摺であるという意味では同じ 1. ルーバー手摺　2. ガラス手摺　のセットや、スクリーンである 3. ルーバースクリーン　4. ガラススクリーン　のセットについても、縦横それぞれ連続上限数が存在する。同様に、ルーバーである 1. アルミルーバー手摺　3. ルーバースクリーン　のセットと、ガラスである 2. ガラス手摺　4. ガラススクリーン　についても縦横それぞれ連続上限数が存在する。

また、ソリューションレイヤーの 6 段目のみは、アルゴリズムによる生成ではなく、手動で作成された「デザイン・パターン」が採用された。これは 6 段目に相当する 8 階は、2 部屋で施主のペントハウスという特殊な用途であったからだ。

本事例では、大きく分けてマップ要素ごとのパネル候補条件、縦方向の連続上限回数、横方向の連続上限数、6 段目の手動「デザイン・パターン」という、4 つの「デザイン・パラメータ」条件によって、「デザイン・パターン」の生成が制御されることとした。さらに、インタラクションの向上を図るため、全ての「デザイン・パラメータ」のはコンピュータプログラム上で GUI(Graphical User Interface) による直感的な操作によって変更可能であり、その変更は「デザイン・パターン」の生成にリアルタイムに反映されるものとした。

2.1.3.3. デザイン・アルゴリズム

まず、マップレイヤーにおける要素配置状態、マップ要素とパネル候補の関係性から、検討対象の 90 個全てのセルにおいて 4 種類のパネルの出現確立が設定される。90 個のセルは、それぞれに設定されたパネル出現確立に従い、一旦ランダムにパネルが割り当てられる。

全てのセルに一旦パネルが割り当てられたら、端から順にパネルの連続数をカウントしていく。まず横方向の連続数を、1. ルーバー手摺、2. ガラス手摺、3. ルーバースクリーン、4. ガラススクリーン　の 4 要素ごと、1. ルーバー手摺　2. ガラス手摺、3. ルーバースクリーン、4. ガラススクリーン　という 2 つのセットごと、1. ルーバー手摺　3. ルーバースクリーン、2. ガラス手摺　4. ガラススクリーン　という 2 つのセットごとという、合計 8 つの数え方でカウントする。

カウントが終了すると、今度はカウントした連続数と横方向の連続数の制限から、連続しすぎているパネルを別の候補に変更する。変更されるパネルの候補も、最初に設定された出現確率に従いランダムで選択される。同様に、今度は縦方向の連続数に対して同様のカウントと、連続しすぎているパネルの変更を行う。

以降、変更されるパネルがなくなるまで、連続数のカウントとパネルの変更を繰り返す。

「デザイン・アルゴリズム」は、なるべくシンプルであるように設定された。これは「デザイン・パターン」の生成速度を追求するためである。結果として、「デザイン・アルゴリズム」が「デザイン・パラメータ」による制限で手詰まりを起こさない限りは、ほぼタイムラグなしで「デザイン・パターン」が生成された。

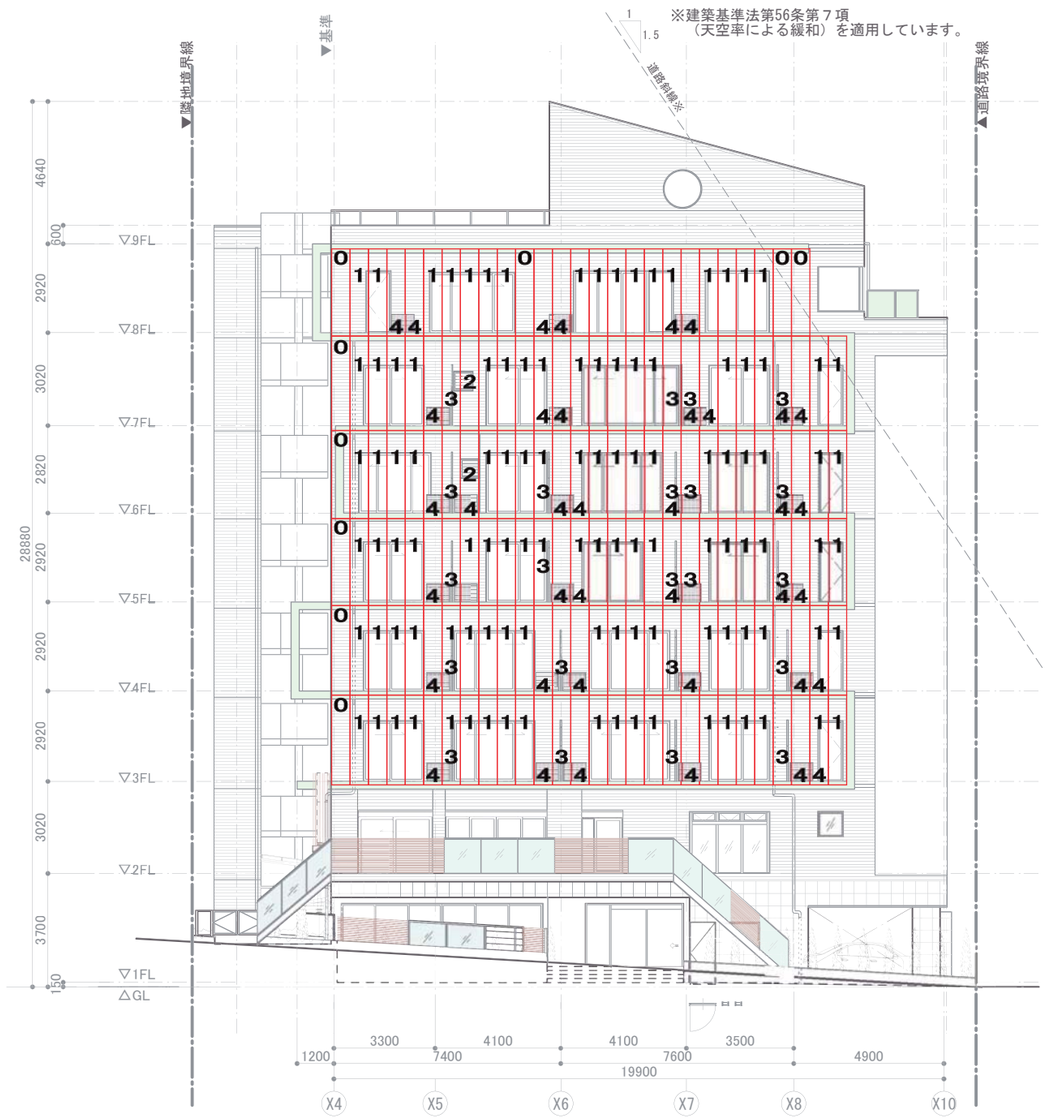
2.1.3.4. デザイン・パターン

本事例においても、コンピュータが高速で生成した「デザイン・パターン」を、人が感性を基準とした選定をおこない、最終候補となる「デザイン・パターン」が決定された。

本事例において、コンピュータによって生成された「デザイン・パターン」が、人の感性によって評価可能であったのは 3 つの理由が考えられる。1 つ目は、これまでの事例からアルゴリズムックな手法において人が有効にインタラクションを得る手法が分かっていたこと、2 つ目が、デザイナーによる感性的要求が「非均質、非偏在」という許容範囲の大きい状態であったこと、そして 3 つ目が、「デザイン・パターン」が均質なグリッド状のセル空間をベースとしていたことである。

しかし本事例が、これまでの事例と最も違う点は、デザインの決定が施主という新しい主体にゆだねられたことである。

マップレイヤへの建築要素の投影



横方向への連続回数制限

	手摺	6 回	スクリーン	2 回
ルーバー　無制限		4 回		2 回
ガラス　無制限		4 回		1 回

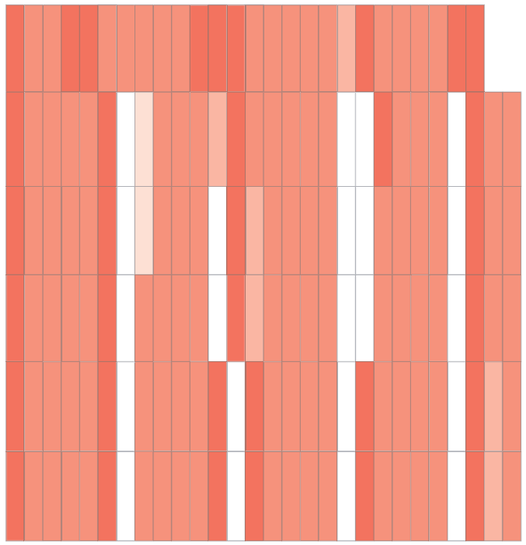
縦方向への連続回数制限

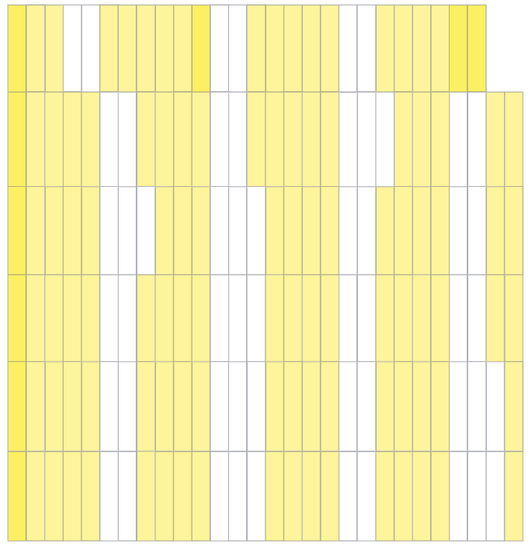
	手摺	無制限	スクリーン	無制限
ルーバー　無制限		4 回		3 回
ガラス　無制限		4 回		3 回

マップ要素とファサードパネルの対応表

	0. RC 壁	1. 掃出窓	2. 腰掛窓	3. 仕切板	4. 室外機
Pattern1	◎	○	△	×	◎
Pattern3	◎	○	○	×	×
Pattern2	◎	×	○	◎	◎
Pattern4	◎	△	△	○	×

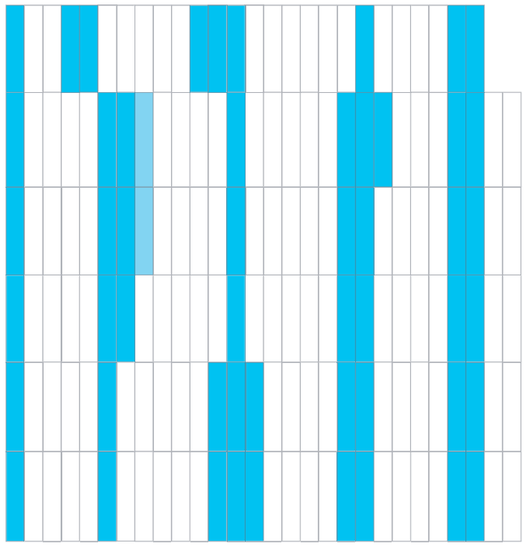
マップ要素との対応によって決まる、ソリューションレイヤにおける、各パネルの出現度

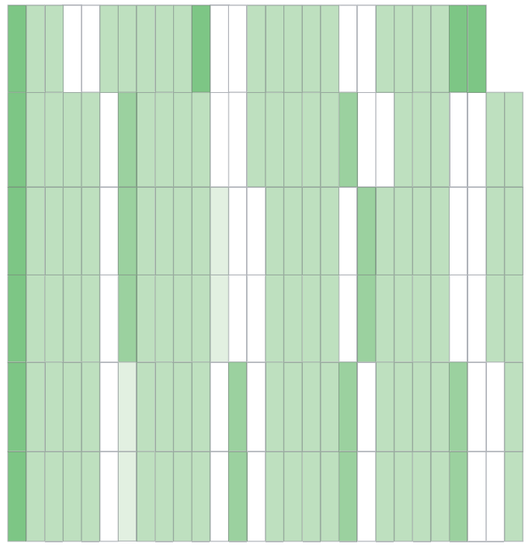




木目調アルミルーバー手摺

ガラス手摺

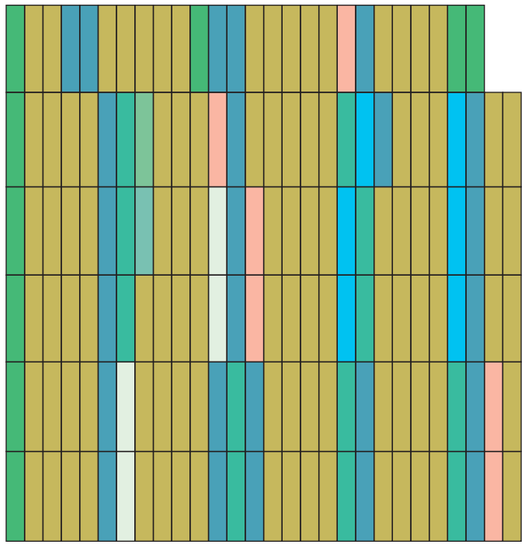




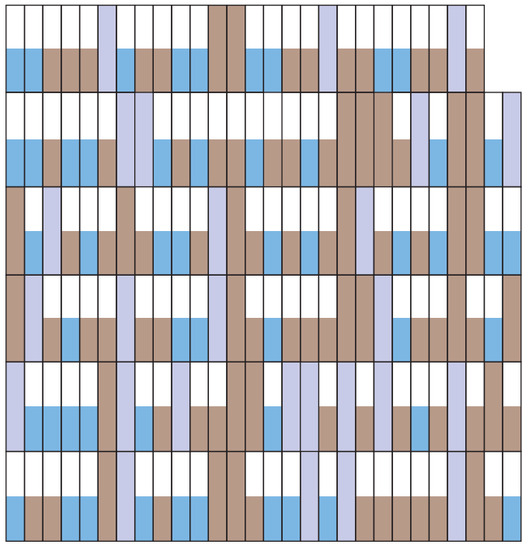
木目調アルミルーバースクリーン

摺りガラススクリーン

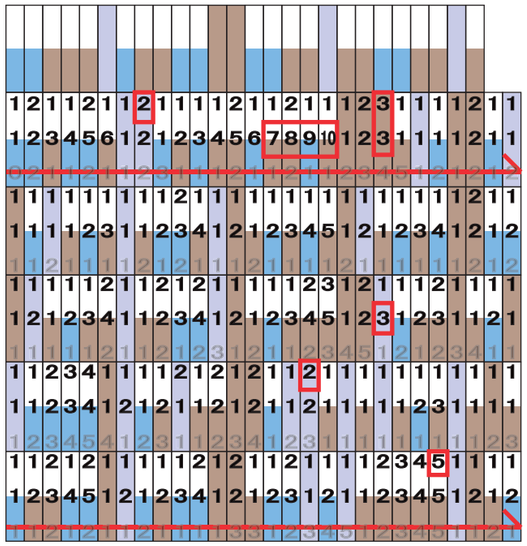
マップレイヤの情報と、マップ要素とパネル候補の関係性から、ソリューションレイヤのセルに生成候補となりうるパネルを設定する。



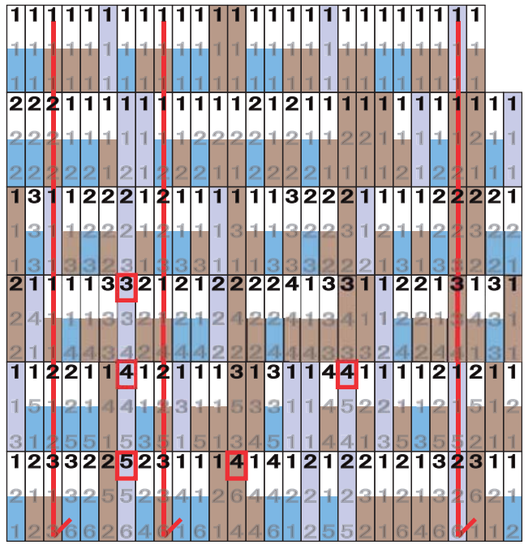
設定された候補パネルの中から、ランダムにパネルを選び、一度ソリューションレイヤのセルに、パネルの配置を行う。



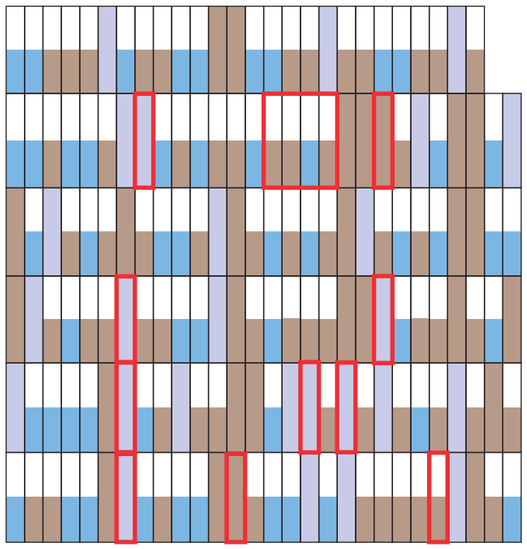
仮に配置されたパネルに対して、同じ要素のパネルが何回連続しているかをカウントする。そして、連続回数の制限を越えている部分をチェックしていく。



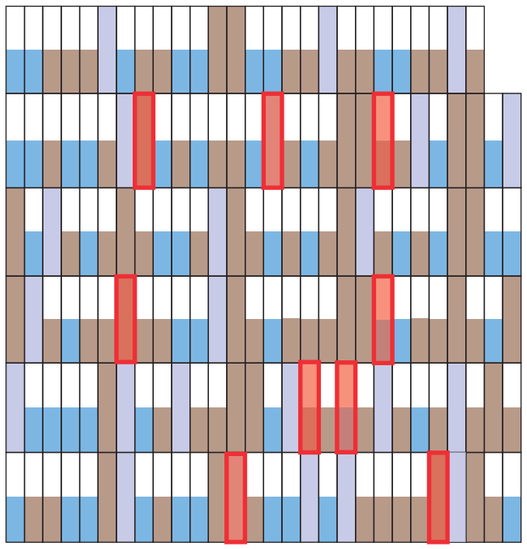
こうしたチェックは、縦方向、横方向の連続に対して、また異なる要素ごとに別々に行われる。



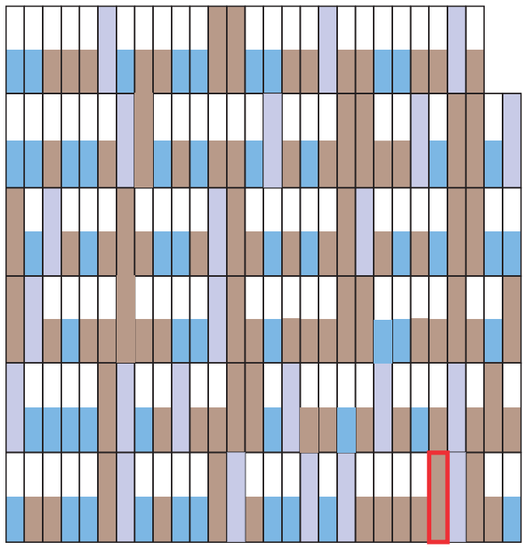
全ての要素の組み合わせ、連続回数制限に対して、チェックが行われ、その結果が集計される。



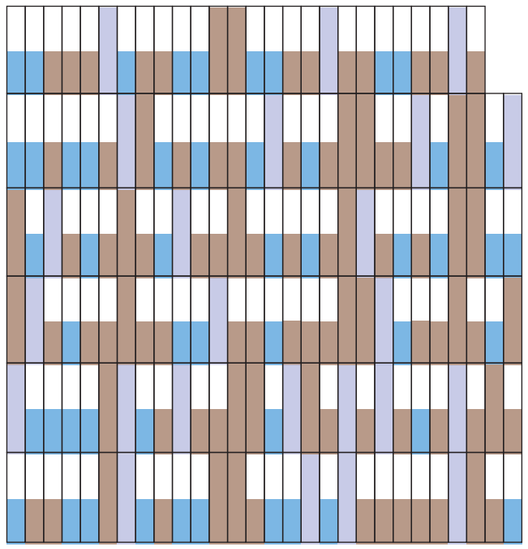
連続回数の制限を越えたパネルを全てピックアップした後、連続制限回数を越えた最初のパネルを別の生成候補に変更する。



再度連続回数のカウントをやり直し、また変更候補をピックアップし、それを変更する。



変更候補がなくなるまで、この処理を繰り返す。

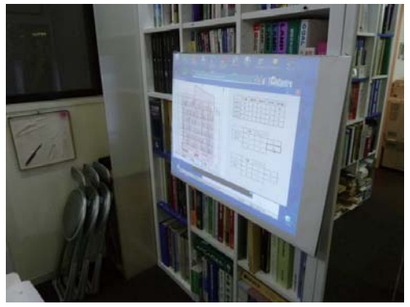


デザインパターンの生成プロセス

2011 年 4 月 22 日に行われた、デザイン決定のための施主との WS の様子と、そこで決定された 3 つでデザイン候補

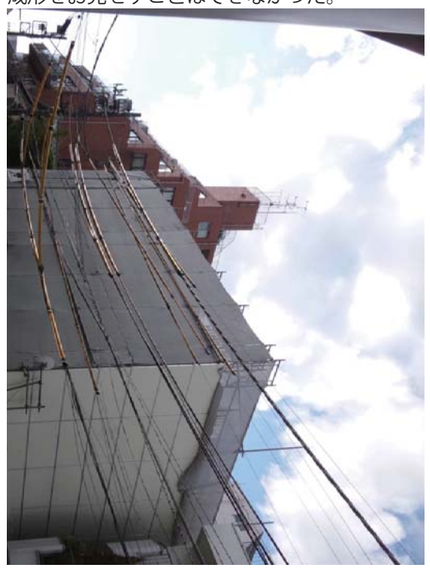
左から著者、IKDS 代表池田靖史さん、連健夫建築研究室代表連健夫さん、そして施主である。

連健夫建築研究室の小川さんから、WS の趣旨とデザインについての説明



プロジェクタによって、コンピュータが生成した「デザイン・パターン」に、皆で意見を出し合う。

ドミエメロードは、現在施工途中であり、完成形をお見せすることはできなかった。



2.1.3.5. 得られた知見

今回のデザイン事例でも、得られたデザイン解は概ね満足のいくものであったといえる。

今回のデザイン対象も、デザイン事例 2 と同様に実現が前提とされているものであった。デザイン課題においても前提となる条件やデザインの目的が明確に存在し、「デザイン・パラメータ」と「デザイン・アルゴリズム」を適切に設定することができた。そして今回の事例でも、「デザイン・パターン」が適切な「デザイン・パラメータ」と「デザイン・アルゴリズム」によって生成されているということ自体が、「デザイン・パターン」の評価対象となっていたため、良いデザイン解を得ることができたと考えられる。またデザイン解に要求された人の感性的な目的が、「非均等、非偏在」の状態という許容範囲の大きいものであったことも、デザイン解に満足ができた理由の 1 つである。

何よりも、今回の事例において特徴的なのが、「デザイン・パターン」からデザイン解を選定していくプロセスに、施主というデザインにおける " 他者 " が参加できたということだ。デザインのプロセスに " 他者 " が参加するというのは、デザイン課題においてもっとも大きな要素であったため、これを可能とする条件が、「デザイン・パラメータ」と「デザイン・アルゴリズム」において積極的に取り入れられた。

今回、施主という " 他者 " が参加したのは、「デザイン・パターン」からデザイン解を選定するというプロセスであった。このプロセスでは、生成結果としての「デザイン・パターン」が重要な要素となる。そのため、施主という " 他者 " が「デザイン・パターン」からインタラクションを得るためには、「デザイン・パターン」の生成が終了するまでの時間をできる限り短くする必要があった。さらに施主という " 他者 " は、デザインにおいては素人であり、デザイナーよりもインタラクションを得るという行為に不慣れであることが予想された。そのため、「デザイン・アルゴリズム」をできる限りシンプルなものにすることで、「デザイン・パターン」が生成完了するまでの時間を短くする必要があった。

「デザイン・アルゴリズム」のシンプルさは、施主という " 他者 " に、デザインの考え方を伝える上でも非常に重要な要素であった。「デザイン・パターン」を判断する上では最低限、このデザインでははは何を実現しようとしていて、どんなことをやっているのかを理解する必要がある。今回はそれを、デザインについては素人である施主という " 他者 " に、それも WS という短い時間で伝える必要があったからだ。

今回もデザイン事例 2 と同様に、「デザイン・パラメータ」と「デザイン・アルゴリズム」の設定によっては手詰まりが発生し、「デザイン・パターン」の生成が不可能である場合が存在した。そのため「デザイン・パターン」から「デザイン・パラメータ」と「デザイン・ア

ルゴリズム」が適切かどうかということについても、判断をする必要があり、「デザイン・パターン」の生成では、その生成結果だけでなく、生成プロセスもビジュアライズされた。

「デザイン・アルゴリズム」をシンプルにするということは、「デザイン・パターン」の生成時間に大きな影響を与えた。結果「デザイン・パターン」の生成完了までの時間は大きく短縮され、「デザイン・パターン」の生成結果から十分なインタラクションを得ることができた。それは、「デザイン・パターン」の生成過程からインタラクションを得るために、わざと「デザイン・パターン」の生成完了までの時間を長くしなければならないほどであった。

今回の事例でも「デザイン・パターン」が生成されたのはセル空間においてであった。このことが、「デザイン・パターン」の複雑性を制限程良く、それを人が評価可能な状態にした。また「デザイン・パターン」が「非均質・非偏在」の状態であるかを評価する際には、似たパネルが過度に連続し過ぎることで均質な印象を与えていないかという、判断基準と判断対象が用いられた。

2.1.4. デザイン事例 4 : 大連大和館・別館 / 内装天井ルーバー

「大連大和館・別館 / 内装天井ルーバー」は、ホテルのロビー空間のための内装天井ルーバーのデザインプロジェクトである。中国の大連において、「大連大和館・別館」というホテルロビーの天井に、複数の木製パネルを組み合わせて、天井全体を覆うようなルーバー状の内装天井をデザインするというものであった。

本事例は、IKDS との共同で行われたプロジェクトである。IKDS が基本設計を行い、すでに施工段階にあった「大連大和館・別館」において、ロビー空間の天井内装となる木製ルーバーの配置パターンデザインに協力したものである。IKDS から提示された内装天井ルーバーの配置というデザイン対象に対し、IKDS と協働でアルゴリズムの検討を行なった、コンピュータへの実装は著者によって行われ、デザインの決定は IKDS によって行われた。

デザイン期間は約 1 ヶ月であった。

2.1.4.1. デザイン課題の整理

本事例のデザイン対象であった天井内装ルーバーは、既に基本的なデザインの考え方が決定されていた。施工が既に完了していた 600mm ピッチの天井下地に対して、天井下地と直行する方向に、木材ルーバーを敷き詰めていくというものであった。

デザイナーは、この内装天井ルーバーが「非均質、非偏在」の状態になるようにしたいという意図を持っていた。そのために、隣り合う木材ルーバーとつなぎ目が一致しないようにしたいということ、2 種類の高さの木材ルーバーを使用したいということなどが要求された。

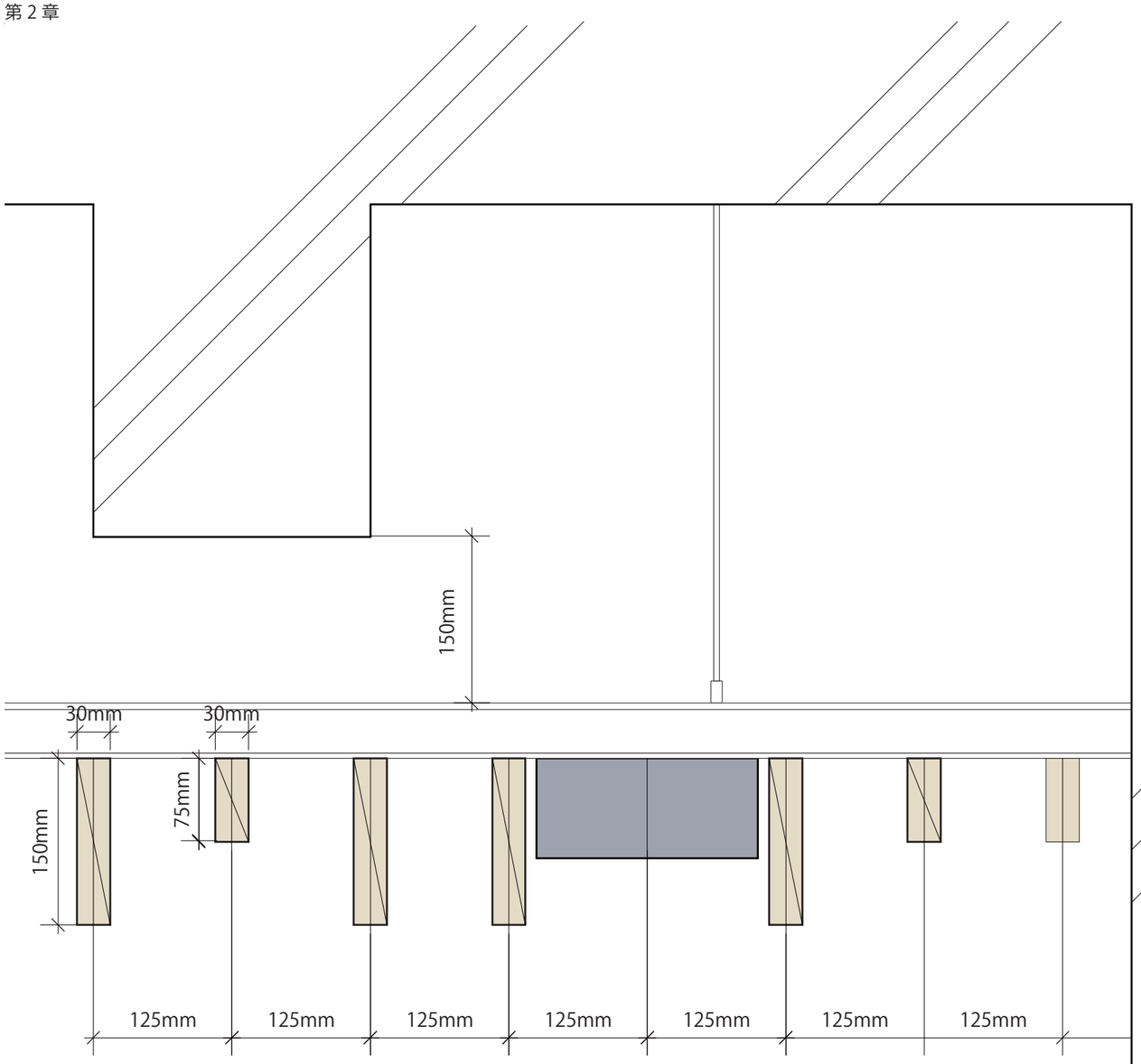
他の要求としては、天井には既に、柱や照明などが設置されており、それらを避けるように木材ルーバーを配置していく必要があった。

しかしこの事例最大の焦点は、中国の大連という現地の施工者の熟練度、技術力に十分な信頼がおけない状況において、このデザインをどのように実現するかということであった。

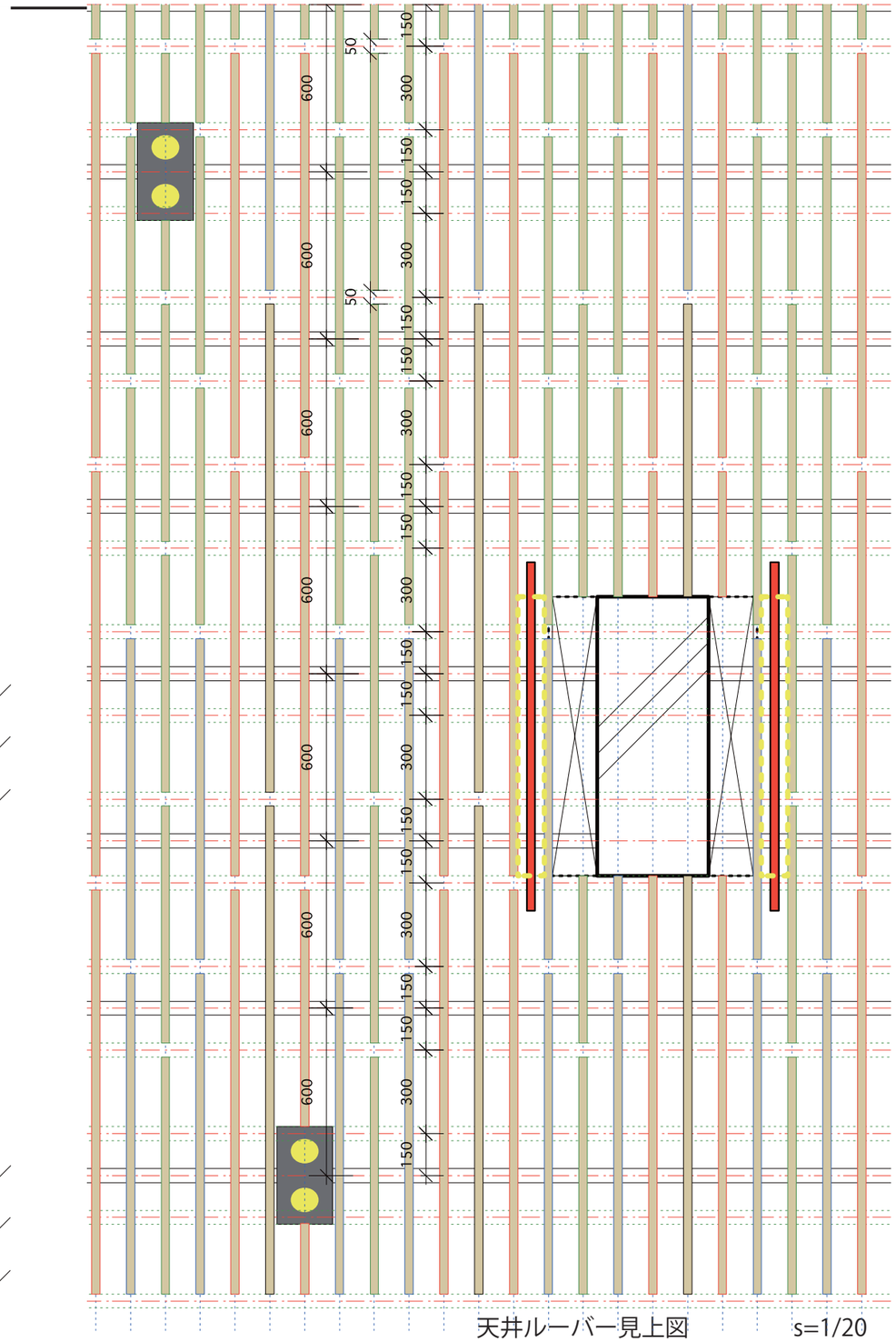
デザイナーは、「非均質、非偏在」という複雑な状態を現地の施工者に伝え、現地の施工者が間違いなくそれを実現することは不可能であると考えていた。また、「非均質、非偏在」の状態をあきらめたところで、既にある柱や照明などによって完全な均質さは望めない。

こうしたデザイン課題に対し、現地では人件費が安く、単純だが膨大な作業の実行コストが低いということに着目し、コンピュータによる「デザイン・パターン」の生成のようなことが、現地の施工者によって実現できないかと考えた。決められた部材バリエーションという「デザイン・パラメータ」と、単純な施工ルールという「デザイン・アルゴリズム」で、「デザイン・パターン」をコントロールしようということだ。

もちろん、コンピュータではなく人が「デザイン・アルゴリズム」を実行する場合、「デザイン・アルゴリズム」は可能な限りシンプルであることが望ましい。また、どのような「デザイン・パラメータ」と「デザイン・アルゴリズム」から、どのような「デザイン・パターン」が生成されるかということを十分に把握する必要がある。そのため、施工者が実行する「デザイン・アルゴリズム」を、コンピュータ上で試めし、デザイナーが意図する「デザイン・パターン」生成する「デザイン・アルゴリズム」と「デザイン・パラメータ」の検討を行った。



天井ルーバー部分詳細断面　s=1/5



天井ルーバー見上図　s=1/20

IKDS	DATE : 2010/05/15 DRAWING : SECTION & CEILING	PROJECT : Dalien Yamato-Kan SCALE : 1/60 (only Layout 1/500)	PLACE : Dalian, Liaoning, China USE : Spring Hotel

2.1.4.2. デザイン・パラメータ

デザイン課題から木材ルーバーの最小長さは天井下地のピッチである 600mm 以上、最大長さは人が扱える 2000mm 以内とされた。また、ルーバーを敷き詰めることを想定すると、長さのバリエーションは 300mm 刻みとなる。本事例では、検討を経た結果、部材の長さは 900mm、1200mm、1500mm、1800mm の 4 種類に決定された。この 4 種類は、1500mm 以上の全ての場合において、2 種類の配置バリエーションがありえる組み合わせとして採用された。隣と部材の継ぎ目が揃わないようにするためには、継ぎ目の場所に複数の選択肢が取れるほうが良い。1800mm の場合は、900mm, 900mm の部材を組み合わせる継ぎ目ありの場合か、1800mm の部材による継ぎ目がない場合の 2 通り、2100mm の場合には、900mm, 1200mm と、1200mm, 900mm の 2 通りといった具合である。

また部材のバリエーションは、この 4 種類の長さのほかに、2 種類の高さという要素がある。2 種類の高さは、照明などの要素との兼ね合いから、75mm と 150mm とされた。この 4 種類の長さ、2 種類の高さという要素からは、最大 8 種類の部材バリエーションが考えられるが、部材のバリエーションを多くすると、施工者が混乱する可能性がある、また「デザイン・アルゴリズム」も複雑にしてしまう可能性があったため、部材バリエーションは 4 種類に限定された。この 4 種類とは、ある長さの部材は、ある高さであるとするもので、やはり検討を経て 900mm × 75mm、1200mm × 150mm、1500mm × 75mm、1800mm × 150mm の 4 種類に決定した。

木材ルーバーは、天井の端から端まで 1 直線に配置され、隣の列との感覚は 125mm とする。

2.1.4.3. デザイン・アルゴリズム

このデザインにおける「デザイン・アルゴリズム」は、かなり単純である。

まず木材ルーバーの配置は、端から 1 列づつ行われる。天井、照明、柱の縁に揃うような部材がある場合は、その部材を優先的に選択して配置する。既に配置された隣の列の継ぎ目と継ぎ目が揃わないように部材を選んで配置していく。これらの条件を満足していれば、部材を配置する順番は自由である。

またこの「デザイン・アルゴリズム」は、実際の施工において実行されるとされていたため、「デザイン・パラメータ」によって「デザイン・アルゴリズム」の手詰まり、それによる配置のやり直しが発生しないようなものが目指された。

この「デザイン・アルゴリズム」は、現地の施工者に対しても十分に伝達可能で、かつ実行可能なほど単純であるが、デザイナーの意図した「デザイン・パターン」を十分実現した。

2.1.4.4. デザイン・パターン

本事例の最大の特徴は、「デザイン・パターン」の生成プロセスにある。たしかに他の事例と同様、コンピュータ上でも「デザイン・パターン」の生成は行われたが、それはあくまで「デザイン・パラメータ」と「デザイン・アルゴリズム」を検討するためであり、そこで生成された「デザイン・パターン」と、実際に実現された「デザイン・パターン」とまったく関係していない。実際の「デザイン・パターン」は、「デザイン・パラメータ」と「デザイン・アルゴリズム」を受け取った現地の施工者によって生成された。

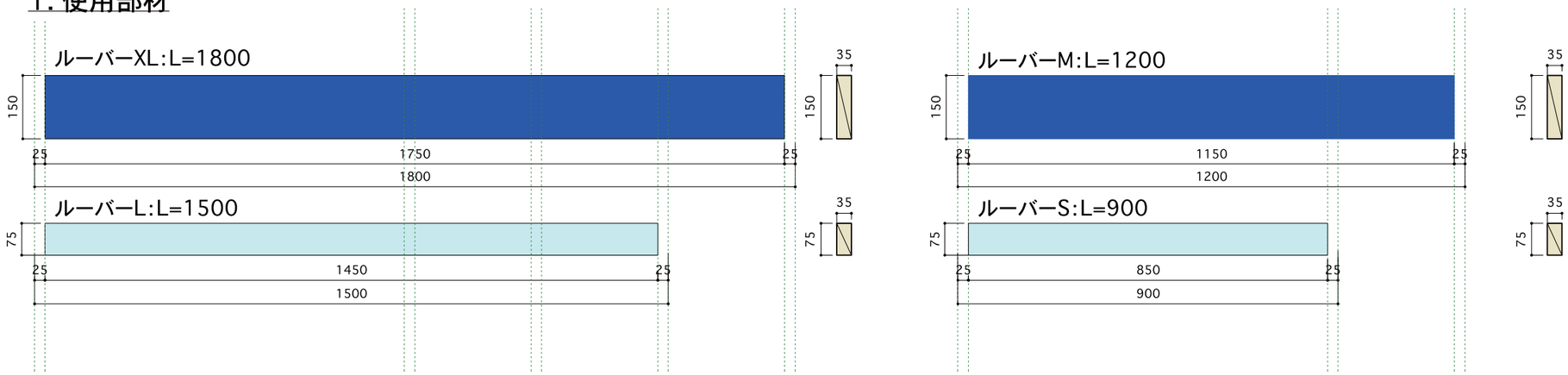
本事例において、コンピュータによる「デザイン・パターン」の生成は、「デザイン・パラメータ」と「デザイン・アルゴリズム」を評価するツールとして使用された。評価は「デザイン・パラメータ」と「デザイン・パターン」が、きちんと「デザイン・パターン」を生成できるか、生成される「デザイン・パターン」がデザイナーの要求を満たすものであるか、という点であった。

デザイナーによる感性的な要求は、他の事例と同様に「非均質、非偏在」の状態を実現すること、その「非均質、非偏在」の状態がデザイナーの感性によってよいと判断できるものであることであった。しかし何度も述べているように「非均質、非偏在」の状態に詳細な優劣をつけることは難しい。そのため目標とされた「デザイン・パターン」は、ある特定のものではなく、曖昧に大きな許容範囲を持って設定され、それを実現するような「デザイン・パラメータ」と「デザイン・アルゴリズム」が決定された。

施工者に施工ルールとして、「デザイン・アルゴリズム」を伝達する際に、
実際に参考にされた資料が以下のようなものである。

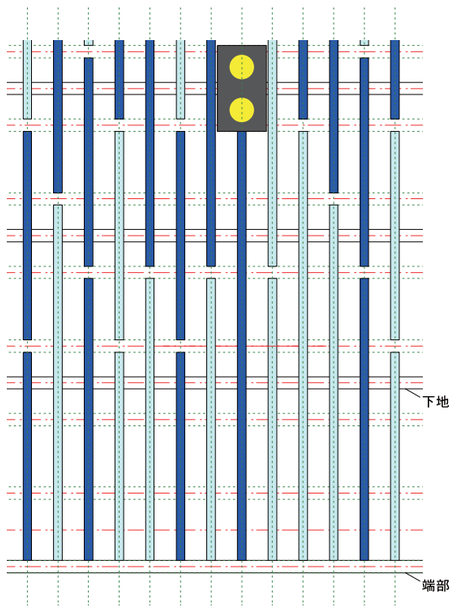
ルーバー天井配置のルールA案

1. 使用部材

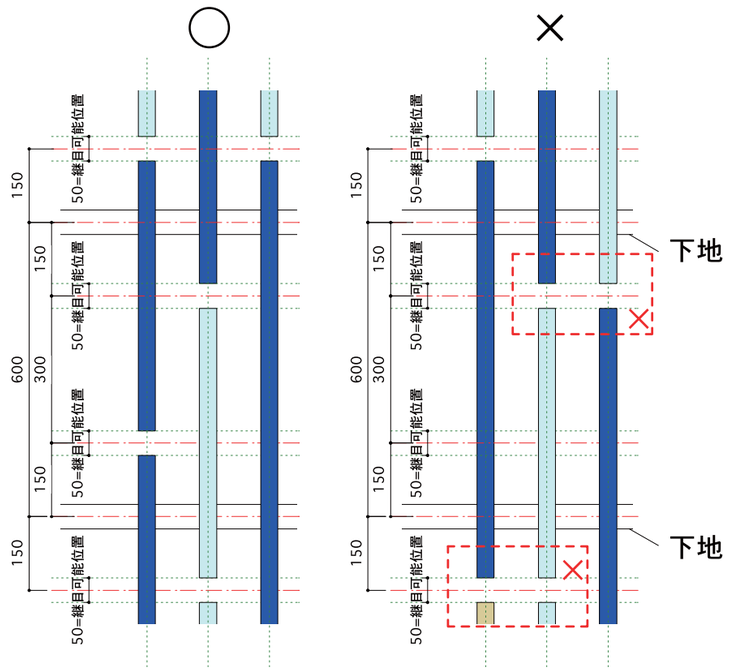


2. 配置のルール

①端部に従う　②なるべく切らない

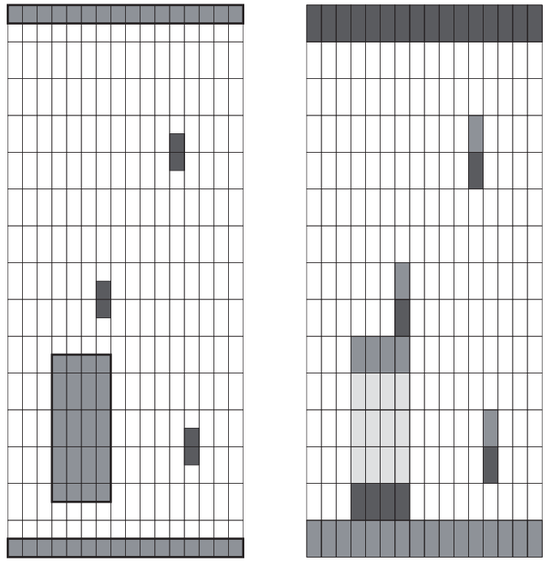


③ルーバーの継ぎ目同士が隣り合わないようにすること。

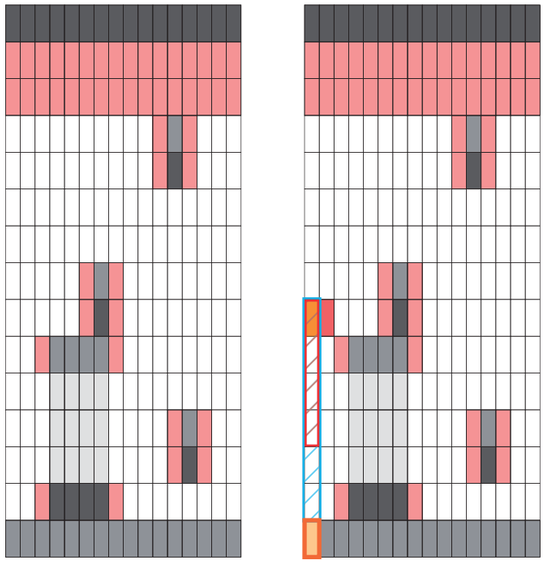


「デザイン・アルゴリズム」の解説

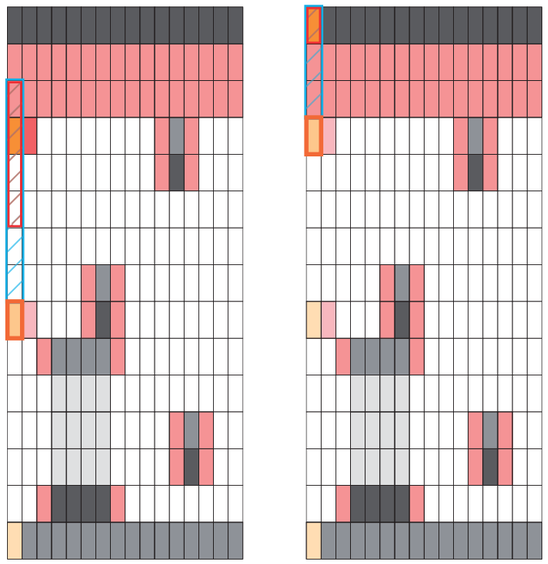
まず、「デザイン・パターン」生成のために、
実際の敷地条件をセル空間に取り込む。



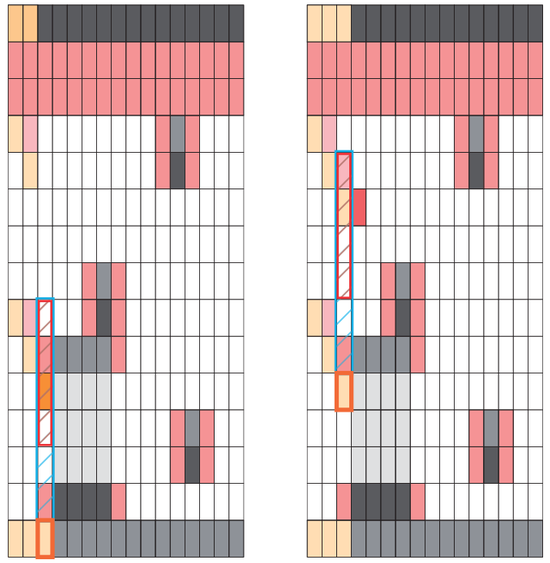
既に継ぎ目が来ることが決定している場所の隣と、継ぎ目がくると手詰まりになってしまうせるに継ぎ目を禁止する情報を与える。端から継ぎ目の配置試行を行っていく。



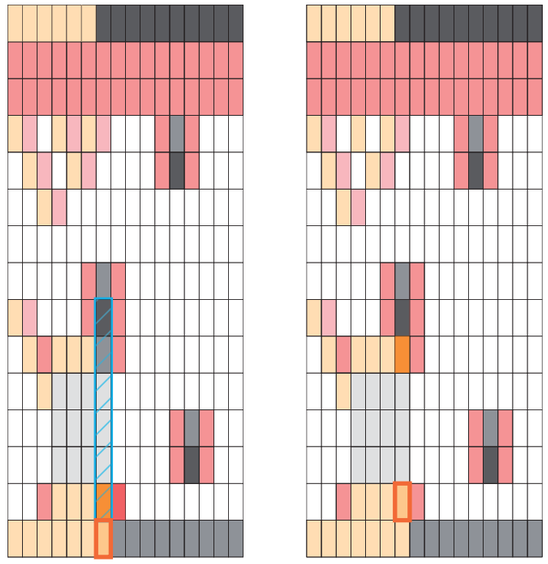
継ぎ目がを配置すると同時に、継ぎ目の隣接セルに継ぎ目を禁止する情報を与える。



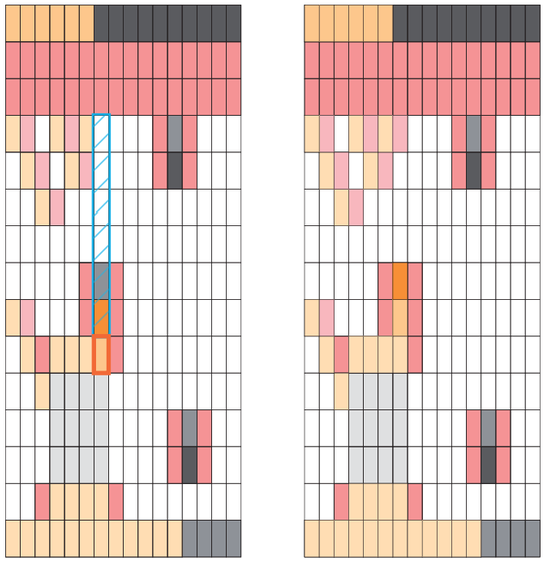
継ぎ目を禁止するセルを避けるように、4 つの部材長さ候補から、次の継ぎ目の位置を選ぶ



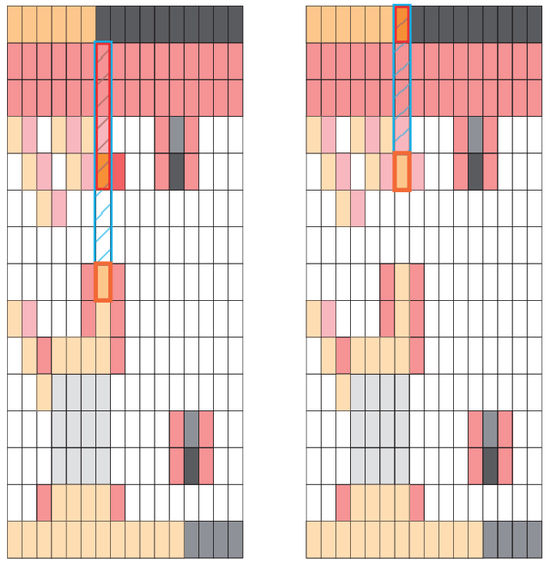
柱や照明の手前には、継ぎ目をあわせない得ればならないため、柱、照明のセルの 1 つ手前には、継ぎ目を強制する情報を持つセルが存在する。



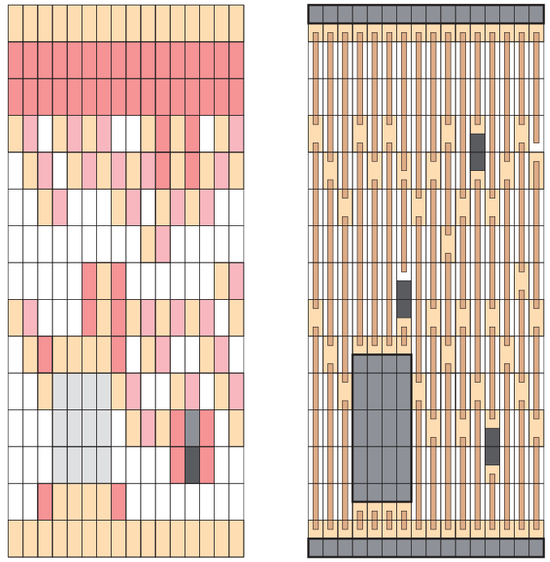
照明や柱にぶつかると、柱や照明の終わりとして設定されているセルに、継ぎ目を配置し、柱、照明の終わりの位置から継ぎ目の生成が再実行されるようにする。



継ぎ目を禁止する条件がいくら重なっても、必ず 1 つは継ぎ目が配置できる場所が残るように、4 つの部材長さは決定されているため、このアルゴリズムには手詰まりがない。

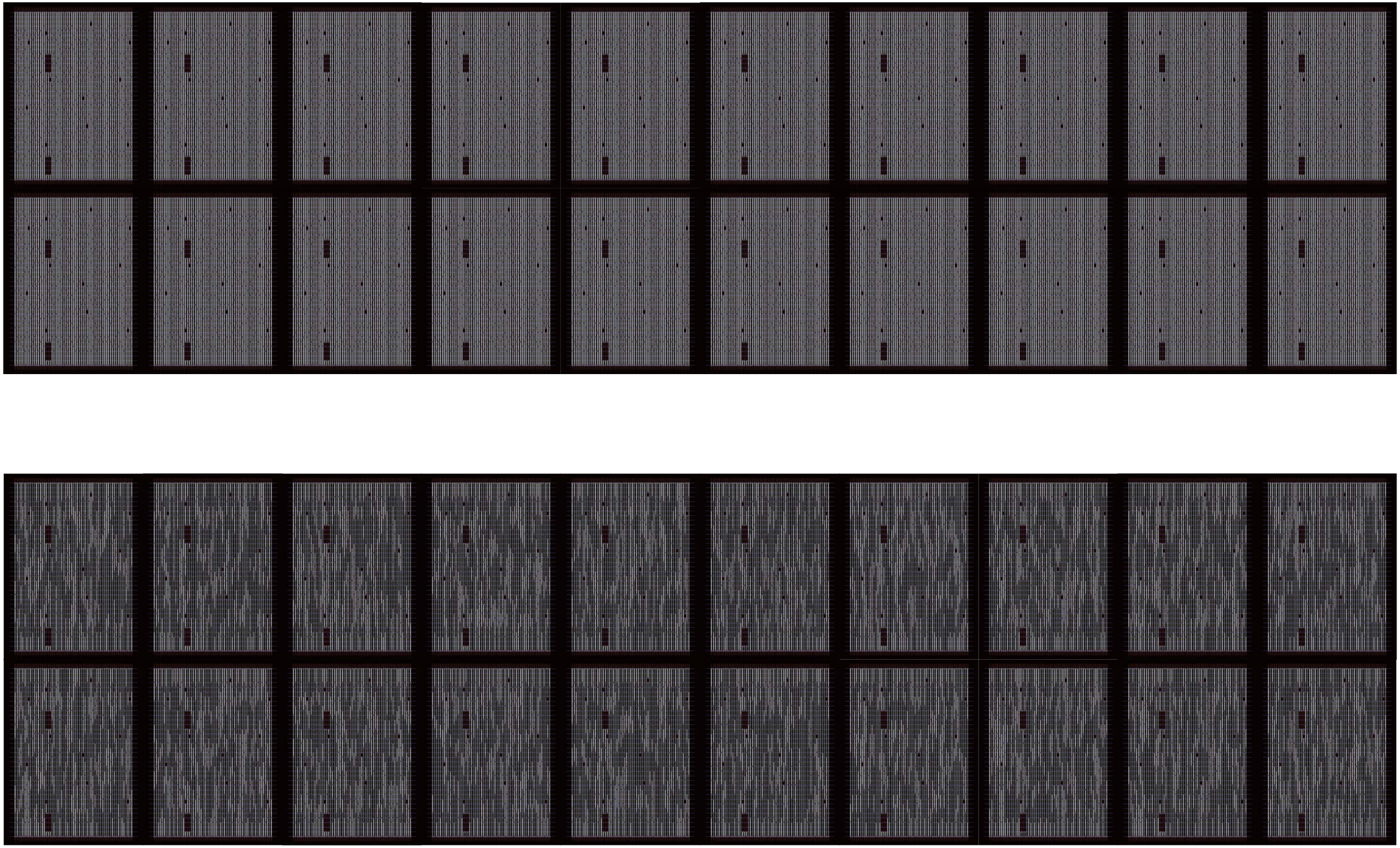


継ぎ目の位置が全て確定したら、それを部材の配置という情報に翻訳する。

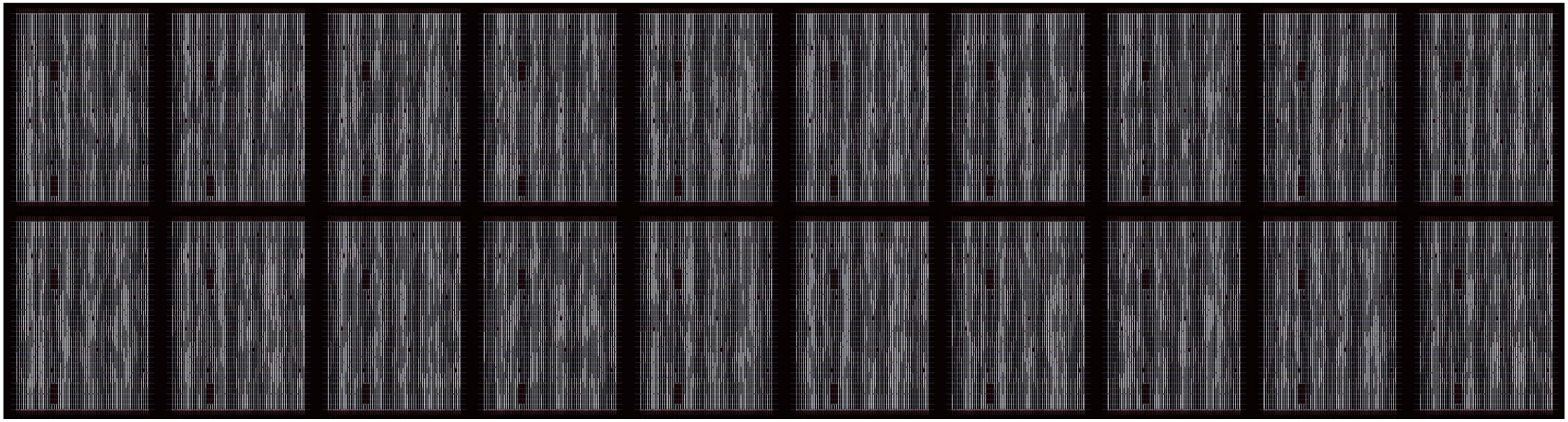


評価対象となった「デザイン・パターン群」の解説

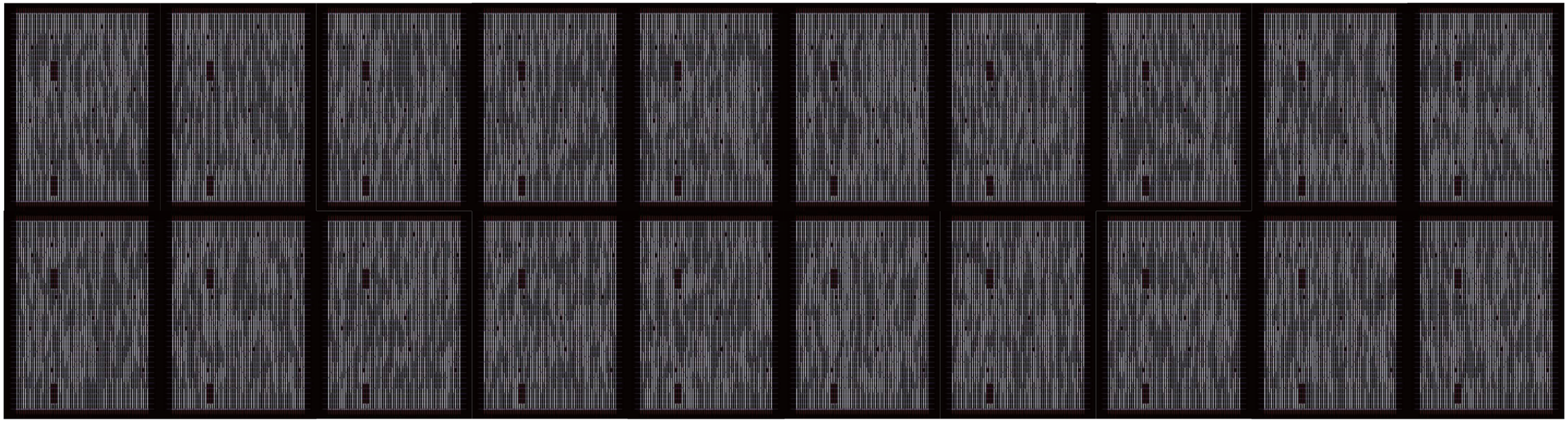
全ての部材が高い場合



900mm 材と 1500mm 材が高く、1200mm 材と 1800mm 材が低い場合



900mm 材と 1800mm 材が高く、1500mm 材と 1800mm 材が低い場合



2.1.4.4. 得られた知見

今回のデザイン事例はかなり特殊である。なぜならば、著者は得られたデザイン解を知らないのである。しかし著者がデザイン解を知らないということが、このデザインが成功したことの何よりの証拠でもある。

今回のデザイン事例においてもデザイン事例 2、3 と同様、デザイン対象は実現を前提としてもので、デザイン課題を具体的に設定することができた。「デザイン・パターン」が適切な「デザイン・パラメータ」と「デザイン・アルゴリズム」によって生成されていることが評価の対象となっていたこと、またデザイン解に要求された人の感性的な目的が、「非均等、非偏在」の状態であったことも同様である。

このデザイン事例が特殊なのは、「デザイン・パターン」の生成を人が、それも施工者という " 他者 " が行ったということである。もちろんコンピュータによる「デザイン・パターン」の生成は行われた。しかしその目的は、デザイン解の候補となる「デザイン・パターン」を生成するためではなく、適切な「デザイン・パターン」を生成するような「デザイン・パラメータ」と「デザイン・アルゴリズム」を模索するためであった。こうして選ばれた「デザイン・パラメータ」と「デザイン・アルゴリズム」は施工者に伝えられ、施工者がデザイン・パラメータ」と「デザイン・アルゴリズム」にしたがって「デザイン・パターン」の生成を行うことで、最終的なデザイン解は得られた。著者がデザイン解を知らないのはこのためである。

このように今回の事例では、アルゴリズミックな手法におけるデザインの目的が、「デザイン・パラメータ」と「デザイン・アルゴリズム」から生成されるよいデザイン解となる「デザイン・パターン」を得ることではなく、よいデザイン解となる「デザイン・パターン」を生成するような「デザイン・パラメータ」と「デザイン・アルゴリズム」を得ることであった。

こうした手法を可能とした理由には、「デザイン・パターン」の評価の仕方がある。

今回の事例において、人が感性による選択を行ったのは、異なる「デザイン・パラメータ」によって生成された「デザイン・パターン」の群であった。「デザイン・パターン」の群は、それを生成した「デザイン・パラメータ」ごとに異なる傾向、雰囲気をもち、人はそれらに感性的優劣をつけた。

「デザイン・パターン」の群に優劣がつけられたのは、この評価には厳密さが求められなかったからである。何故ならこの「デザイン・パターン」の群は、あくまで評価のために生成されたものであり、実際にデザイン解となる「デザイン・パターン」と同じ雰囲気、傾向を持つに過ぎない。実際にデザイン解となる「デザイン・パターン」は施工者によって生成される別のものであるため、この「デザイン・パターン」の群を厳密に評価することは意味の無い。

また今回の事例では新しい評価として、「デザイン・パラメータ」の制限によって「デザイン・アルゴリズム」が手詰まりを起こさないかどうかが判断された。デザインプロセスに施工者という " 他者 " が介入することから、「デザイン・アルゴリズム」は今まで以上にシンプルであることが求められた。これまでは、「デザイン・アルゴリズム」のシンプルさとは「デザイン・アルゴリズム」における判断内容のシンプルさであった。今回はそれに加え、手詰まりを起こさないという「デザイン・アルゴリズム」の挙動のシンプルについてが追求された。

コンピュータのような高速な処理能力を持つものが、仮想の空間上で「デザイン・パターン」の生成を行う場合は、やり直しや手詰まりはさほど問題にならない。しかし「デザイン・パターン」の生成を人が実空間において行うような場合、やり直しや手詰りは「デザイン・パターン」の生成実行コストを大きくあげてしまうからである。

「デザイン・アルゴリズム」が手詰まりを起こさないようにするためには、それにある程度の自由度が求められる。また「デザイン・パターン」の生成を人が行う場合において、「デザイン・アルゴリズム」の自由度は、人の気まぐれや失敗を許容する部分として有効に働く。

そしてデザインプロセスに施工者という " 他者 " が介入することから、「デザイン・アルゴリズム」は今まで以上にシンプルであることが求められた。

今回の事例では、「デザイン・パターン」の生成プロセスにおいて、「デザイン・アルゴリズム」の追加が行われた。これは、施工者が「デザイン・パターン」を生成中に、新しい追加のデザイン条件を発見し、それを解決するために行ったものである。これが可能だったのは、「デザイン・アルゴリズム」がシンプルであり、かつ自由度が確保されるように設定されていたからである。

この「デザイン・アルゴリズム」の追加という行動は、アルゴリズミックな手法によるデザインに新しい可能性を提示してくれる。アルゴリズミックな手法によるデザインが、適応を必要とするデザイン課題に答えることができるのは既に知られていた。しかしこれは、固定された「デザイン・アルゴリズム」において、生成される「デザイン・パターン」が、条件の変化に対して受動的に対応可能だということに過ぎなかった。

しかし、最初に設定された「デザイン・アルゴリズム」が、十分にシンプルで、かつ自由度を持っていれば、「デザイン・パターン」の生成途中であっても、「デザイン・アルゴリズム」を追加し、条件の変化をより積極的に解決するということが可能であるということを示している。

2.2. アルゴリズミックなデザイン事例を通じた考察

これまでは個別のデザイン事例において、アルゴリズミックな手法がどのように構築され、それが各事例を通してどうに継続、展開されてきたかについてを述べてきた。ここからは、そうして各事例から得られた知見を通して見る事で、アルゴリズミックなデザイン手法そのものに対する分析を行い、2 章のまとめとする。

アルゴリズミックな手法では、人の感性によって強く要求されたデザイン課題に対し、それを満たすようなデザイン解を求めることはできない。また具体的ではデザイン課題が無い場合も、同様にデザイン解を求めることはできない。しかし、これはある意味当然といえる。そもそも、デザイン解に対して強く要求する感性的条件があるのならば、その感性を元にデザインを形にするべきである。コンピュータと協働できる、アルゴリズミックな手法を用いる必要はそこにはない。そしてアルゴリズミックな手法によるデザインとは、デザインを問題解決のプロセスであると捉え、この解を求めるという手法である。デザイン課題が具体的でないということは、取り組むべき問題も分からないということであり、これも論外である。

アルゴリズミックな手法によるデザインとは、既に実現したいデザイン解が大まかには想定されているものの、それを具体化するためには解決しなければならない具体的なデザイン課題が存在するようなケースにおいて、想定されたデザイン解の形から大きく外れることなく、かつデザイン的な課題を解決しているという具体的なデザイン解を模索する手段である。

その一例が、「非均質、非偏在」な状態を実現したいといったデザイン課題なのである。

4 つのデザイン事例の内、デザイン事例 1 を除く 3 つの事例において、「デザイン・パターン」の生成空間としてセル空間が採用された。そしてセル空間ではな 9 自由度が高い空間を「デザイン・パターン」の生成空間として採用したデザイン事例 1 だけが、よいデザイン解を得ることができなかった。しかしこれは、アルゴリズミックな手法がセル空間に生成された「デザイン・パターン」でしか、よいデザイン解を提示できないということではない。

アルゴリズミックな手法によるデザインでよいデザイン解を得るためには、「デザイン・パターン」からデザイン解を決定する過程において、人の感性による判断が必要になる。しかし、アルゴリズミックな手法で生成される「デザイン・パターン」は複雑であることが多く、それを人が厳密に感性で評価することは不可能である。

アルゴリズミックな手法においてセル空間がよく使用されるのは、セル空間では要素の位置と隣接関係が明確なであり、この空間では複雑さが程良く制限されるからだ。これは「デザイン・パターン」を生成する際にも、評価される際にも言えることである。

ここで重要なのが、複雑さが程良く制限されているということである。デザイン事例を通してまとめられた、評価対象となる「デザイン・パターン」の複雑さを制限する方法を用いることで、セル空間という手法に限らずとも、アルゴリズミックな手法によるデザインによってよいデザイン解を得ることができると考えられる。

「デザイン・パターン」の複雑さを制限する方法の 1 つは、人が「デザイン・パターン」のどういう点を評価すればよいかを明確にするということである。たとえ評価対象の「デザイン・パターン」が複雑なものであった場合でも、その度の部分を、どのように評価すればよいかが明確であれば、人はそれを評価することができる。

2 つ目は、評価対象となる「デザイン・パターン」の複雑さを、評価段階においてある程度限定した形で提示するということである。

そして最後が、人による評価が、厳密、正確でなくてもよいような仕組みを用意するということである。これはデザインプロセスに何らか別の仕組みや考え方を用いることで、デザイン解の決定にとって、「デザイン・パターン」を感性的に評価することの重要性が低くても構わないようにしたり、感性的に評価が厳密に 1 つの最も優れた 1 つの「デザイン・パターン」を選び出すといった厳密な感性的判断を必要としないようにするということである。

アルゴリズミックな手法によるデザインもともとは、人のデザインプロセスにコンピュータという " 他者 " が参加するための手法である。しかしアルゴリズミックな手法によるデザインでデザイン参加できる " 他者 " は、コンピュータだけではない。今回参照した事例においても、これまではデザインに参加をすることのできなかった、施主や、施工者という " 他者 " がデザインに参加し、デザイン解となった「デザイン・パターン」の生成に大きな役割を果たした。これまで、こうした " 他者 " はデザインに関係する存在でありながら、デザイナーを介することでしか、デザインに参加をすることができなかった。しかし、デザインにおいては重要な存在であり、彼らを抜きにしてはデザインそのものが成り立たなくなってしまう。そういった " 他者 " が、アルゴリズミックな手法によって直接的にデザインに参加することで、デザイン解をより良いものとする新しい視点が提示される。そしてこのように、アルゴリズミックな手法がデザイン参加を可能とする、デザインにとって重要で、有用な " 他者 " は、まだまだ存在すると考えられる。

アルゴリズミックな手法において「デザイン・パターン」は、人による評価の対象とされている。しかしこの評価には、2 つの目的が存在する。1 つは、「デザイン・パターン」からデザイン解を選定すること、もう 1 つが「デザイン・パターン」から「デザイン・パラメータ」と「デザイン・アルゴリズム」が適切であるかを判断することである。

前者は、デザイン解である「デザイン・パターン」を「デザイン・パラメータ」と「デザイン・アルゴリズム」から複数生成し、それらの中から人の感性によって最も優れていると判断できるもの、よいデザイン解を選ぶことである。ここで「デザイン・パターン」は評価対象であると同時に、選択対象であり、デザイン解の候補である。

後者は、「デザイン・パターン」の生成プロセスから「デザイン・パラメータ」と「デザイン・アルゴリズム」が適切であるかを判断し、よいデザイン解である「デザイン・パターン」を生成する「デザイン・パラメータ」と「デザイン・アルゴリズム」を選択することである。ここでは、「デザイン・パターン」は評価対象であり、選択するためのツールであり、デザイン解のモックアップである。

デザインとは、ヴィジュアライズであり、アウトプットである。そういう意味では、前者のアルゴリズム「で」デザインするプロセスは、確かにデザインにとって重要なプロセスである。しかしアルゴリズミックな手法によるデザインにおいて、そのデザインの多くの部分を占め、アルゴリズム「で」デザインするプロセスよりも遥かに重要性が高いのが、後者のアルゴリズム「を」デザインするプロセスなのである。

「デザイン・アルゴリズム」は、デザイン課題を十分解決できる範囲においてシンプルで、かつある程度の自由度があったほうがよい。

「デザイン・アルゴリズム」がシンプルであれば「デザイン・パターン」の生成時間が短縮され、人が「デザイン・パターン」の生成からインタラクションを得られるかどうかに影響を与える。アルゴリズミックな手法によるデザインでは、人が「デザイン・パターン」を操作することができず、それは「デザイン・パラメータ」の操作によって間接的に行われる。そのため、人が「デザイン・パターン」の生成からインタラクションを得られなければ、デザインを進展させていくことができない。

またアルゴリズムが十分にシンプルであり、人が「デザイン・パターン」の生成から容易くインタラクションを得られる状態が実現できれば、「デザイン・パターン」からデザイン解を選定するプロセスに " 他者 " の参加することが可能になる。

さらに、アルゴリズムが十分にシンプルであれば、アルゴリズムにしたがって「デザイン・パターン」を生成するプロセスにも " 他者 " が参加することができる。

「デザイン・アルゴリズム」の自由度も「デザイン・パターン」の生成速度と、「デザイン・パターン」の生成するプロセスに " 他者 " が参加することを容易にする。「デザイン・アルゴリズム」にある程度の自由度が保障されているということは、「デザイン・パターン」の生成プロセスにおいてミス、気まぐれ、不測の事態が許容され、また多くの選択肢が用意されているということだからだ。

自由度を持つ「デザイン・アルゴリズム」は、生成される「デザイン・パターン」のバリエーションを広げる。「デザイン・パターン」のバリエーションが広がれば、その中、人では生成できなかった、人の感性的に優れているといった、よいデザイン解が含まれる可能性が増す。

3. アルゴリズム的な手法による実験的デザイン

3.1. デザイン対象の設定

ここでのデザイン実験ではデザイン対象として、デザイン事例 1 と同じ戸建住宅街における住宅建築群の配置計画を採用した。

デザイン対象をデザイン事例 1 と同じものに設定したのにはいくつかの理由が存在する。

1 つ目の理由は、都市デザインというデザイン対象がアルゴリズミックな手法によるデザインと相性がよいということである。都市は多くの要素によって構築されるデザイン課題である。また、長いタイムスパンで計画、建設される都市は、環境の変化などに適応的であることも必要とされる。

2 つ目の理由が、デザイン対象をすでに 1 度扱ったことのあるものにすることで、デザイン対象に関する調査・研究の手間を軽減しつつ、デザイン対象から十分な知見を得ることができるからである。また他のデザイン過程についても、既に 1 度目のデザイン事例を生かし、より良いデザイン実験ができると考えたからだ。

3 つ目が、前回のデザイン解が満足のいく物でなかったというということがある。既に 1 度デザイン解を求めたデザイン課題に、再度取り組むということに意味があるのかどうかという問題は当然あったものの、前回のデザイン解がそれほどよいものではなかったため、新しいデザイン解を求める意味と余地が有ると考えた。

4 つ目が、この研究全体の検証ができると考えたからである。デザイン事例 1 は、最初に取り組んだデザイン事例である。その後 3 つのデザイン経験を経て得られた知見をもって、あえて最初の事例と同じデザイン対象に取り組み、2 つの結果を比べることで、著者が 4 つの事例で得た知見とそこから考察されたアルゴリズミックなデザイン手法についての考え方を検証することができると考えられた。

ただしデザイン事例 1 では、配置される建築の用途を同一規模の戸建住宅に限定していた

が、今回の実験では規模が異なる戸建住宅、戸建住宅以外の規模、用途の建築の配置も想定されるものとした。また、敷地の設定のデザイン事例 1 と同じものとするが、デザインが適応的であるかどうかの評価のために、別の敷地において実験をする必要もあると考えられる。

3.2. デザイン対象に関する調査・研究

デザイン事例 1 の際に行った研究より、良いとされる住宅街計画の事例を集めその手法を分析すると、既存の住宅街計画手法は、以下のような絶対的に満たされなければならない条件、

1). 各住戸へ車でのアクセスが可能である
2). 各住戸にライフライン設備が通せる
3). 私有の領域が明確に定まっている
4). 雨水排水システムが備わっている

と、以下のようなある許容範囲内の値が満たされなければならない条件、

5). 各住宅の面積規模が一定範囲内である
6). 各住戸へのアクセス距離が一定範囲内である
7). 各住戸の日照が一定範囲内である
8). 住宅密度が一定範囲内である
9). 公共空地の規模と設置頻度が一定範囲内である
10). 各住宅間の隙間が一定範囲内である

に注視していることを見て取ることが出来た。また以上の条件に加え、以下のような条件、

11). 各住宅が良質な隙間・空地によって関係している

を満たすことによって、良質な空間性をもつ住宅街というものが計画されていることが分かった。下記はそれぞれの条件が、トップダウン的住宅街計画のどのような操作によって保障されているかという対応関係をしました表である。

こうした条件を満たすための手法として、既存の計画手法では、住宅地の敷地を道路と住宅敷地境界により、グリッド状に等分割することから始まる、トップダウン的な手法を用いる。この手法は確かに、上記条件を効率的に満たすことができるものであるが、その一方で前述した既存住宅街の問題の原因になってしまっているとも言える。それは、住宅街が質の良い空間性を獲得するための条件である 11). に対して、このシステムが何ら解答を示せていないことにも関係している。既存の手法では、このシステムに基づいた全体の構成が決まった後に、部分のデザインとして空間的関係性の計画がなされる。

	既存手法が、道路のパターンで解決している条件		既存手法が、敷地境界のパターンで解決している条件
	直接的	間接的	
各住戸に車でのアクセスが可能である	○		
各住戸にライフライン設備が通せる		○※1	
私有の領域が明確に定まっている	○※2		○※2
雨水排水システムが備わっている		○※1	
各住宅の面積規模が一定範囲内である	○※2		○※2
各住戸へのアクセス距離が一定範囲内である	○		
各住戸の日照が一定範囲内である	○※2		○※2
住宅密度が一定範囲内である	○※2		○※2
公共空地の規模と設置頻度が一定範囲内である	○		○※2
各住宅間の隙間が一定範囲内である	○※2		
各住戸が良質な隙間・空地によって関係している			

※1　道路上あるいは道路地下が、設備導線となっているため
※2　道路と敷地境界が、共にグリッド状の分割境界線となっているため

これらの条件は、別の観点からの再整理が可能である。

それはこれらの条件を、住宅街が全体としての機能を構成するために必要とされることと、住宅建築の集積として各住宅建築同士が良い空間的関係性を持つために必要とされることとに分類するということである。この 2 つの条件は、前者の方が比較的システマチックな条件であり、後者の方がより感性的判断を必要とする条件であるといえる。また思考のスケールとしても、前者は広域における多くの要素が関係してくる条件であり、後者は限定された領域における限られた要素についての条件である。これは後に、アルゴリズムにおいて人とコンピュータそれぞれの思考を、どのように分けつつ繋ぐかということを考える上で重要になる。

	住宅街全体の機能を構成するために必要とされる条件	住宅建築が群として良い状態を構成するために必要とする条件
各住戸に車でのアクセスが可能である	○	
各住戸にライフライン設備が通せる	○※1	
私有の領域が明確に定まっている		
雨水排水システムが備わっている	○※1	
各住宅の面積規模が一定範囲内である	○	○
各住戸へのアクセス距離が一定範囲内である	○	
各住戸の日照が一定範囲内である	○	○
住宅密度が一定範囲内である	○	
公共空地の規模と設置頻度が一定範囲内である	○	○
各住宅間の隙間が一定範囲内である		○
各住戸が良質な隙間・空地によって関係している		○

※1　1). の道路とアクセスの関係によって自動的に解決される条件である

3.3. デザイン事例からのフィードバック

住宅街において必要とされる条件を再整理した結果、住宅街におけるデザイン課題はが、住宅街全体という広域で多くの要素に関係する機能をシステマティックに解決する部分と、住宅建築が小規模な群として満たすべき感性的な条件を解決すべき部分からなり、かつこの 2 つ独立に解決されるのではなく繋がっていることが必要となることが分かった。

デザイン事例 1 では、この 2 つの部分の違いを認識しないまま、両者を同時に解決できるようなアルゴリズムの検討が行われた。しかしこの 2 つの部分を同時に解決することは難しく、結果アルゴリズムはどちらの問題も中途半端に扱うようなものとなってしまった。ここで改めて、両者を見比べてきると、前者はどちらかといえばコンピュータが解決を得意とする部分、後者は人が解決を得意とする部分である。デザイン事例 1 のアルゴリズムの問題点は、こうした違いを認識せず、両者をコンピュータによる解決で統合しようとしたことにあると考える。

デザイン事例 1 ではさらに、人が感性的に判断すればよい問題の限定も十分でなかった。コンピュータが生成した住宅の配置パターンに人が感性的な優劣をつけることで、デザイン解を模索するという大まかな枠組みはあったものの、では人が何をどのように感性的に判断すればよいのかということについて限定がされていなかった。

3.4. アルゴリズミックな手法の検討

デザイン対象への調査、研究とこれまでのデザイン事例からのフィードバックから、住宅街における建築群の配置デザインというデザイン課題は、スケールが小さく感性的な判断を必要とする部分のデザインと、規模が大きく機能的なシステマティックな性能を要求される全体のデザインを、どのように一体的な手法で解決するかが焦点となる。当然、前者は人、後者はコンピュータが得意とする問題であるが、それぞれが別々に解決されてしまうのでは、既存計画手法と同様の問題を抱えてしまうことになる。そこで今回は両者を繋ぐものとして、アルゴリズムが機能できるのではないかと考えた。

部分のデザインは、人がアルゴリズムに従い感性的な判断によって行い、全体のデザインは、コンピュータがアルゴリズムに従いシステマティックに行い、それを人が判断する。部分と全体それぞれを、適切に解決できる別の思考方法によって行いながらも、それぞれをアルゴリズムが繋ぐことで 1 つのデザインとして統合できると考える。

アルゴリズムは、部分のデザインにおいて、人がある自由度をもって感性なデザイン探索が行えるということを保障しつつ、全体のデザインにおいて、そうした部分の集積が全体の

機能を損なうことがないように制御するように機能する。アルゴリズムにおいて前提条件とするべきなは、部分と全体の両方に影響を与える条件である住宅面積とその日照条件が適切であると考える。

3.5. 想定される具体的なデザイン

これらのことから、具体的なデザイン手法として下記のようなものが想定される。

まず、「デザイン・パラメータ」の設定である。それは敷地条件と、配置される建築の形状、建築の配置位置に関わる制限条件である

敷地内の建築の建築面積と延べ床面積を一定の範囲内に収めるため、建築を縦横高さに一定制限のある直方体のボリューム内に収まるように計画させる。この直方体のボリュームは建築概形と呼び、そのボリュームは建築の用途によって異なる制限を受ける。建築概形のボリュームに収まる範囲内であれば、建築個々の計画は自由であるとする。

各建築への日照条件を一定にするために、建築概形には正面となる面が設定され、建築概形の向きはその面が真南から傾いてはいけない制限角度として 45 度が設定されている。また建築概形には、他の建築が配置されることを許可しない排他敵領域が付属する。これは、建築概形の大きさと向きに従い、自動的に生成される領域であり、その形状は日影時間図を参考に決定された。これは建築への日照条件を一定にするためと、建築の周辺に適切な空地、道路や庭となるような空地が確保されるようにするためである。

建築の配置位置は、既存住宅からの距離によって制約を受ける。この制約には、近づいてはいけない最小距離と離れてはいけない最大距離が設定されている。

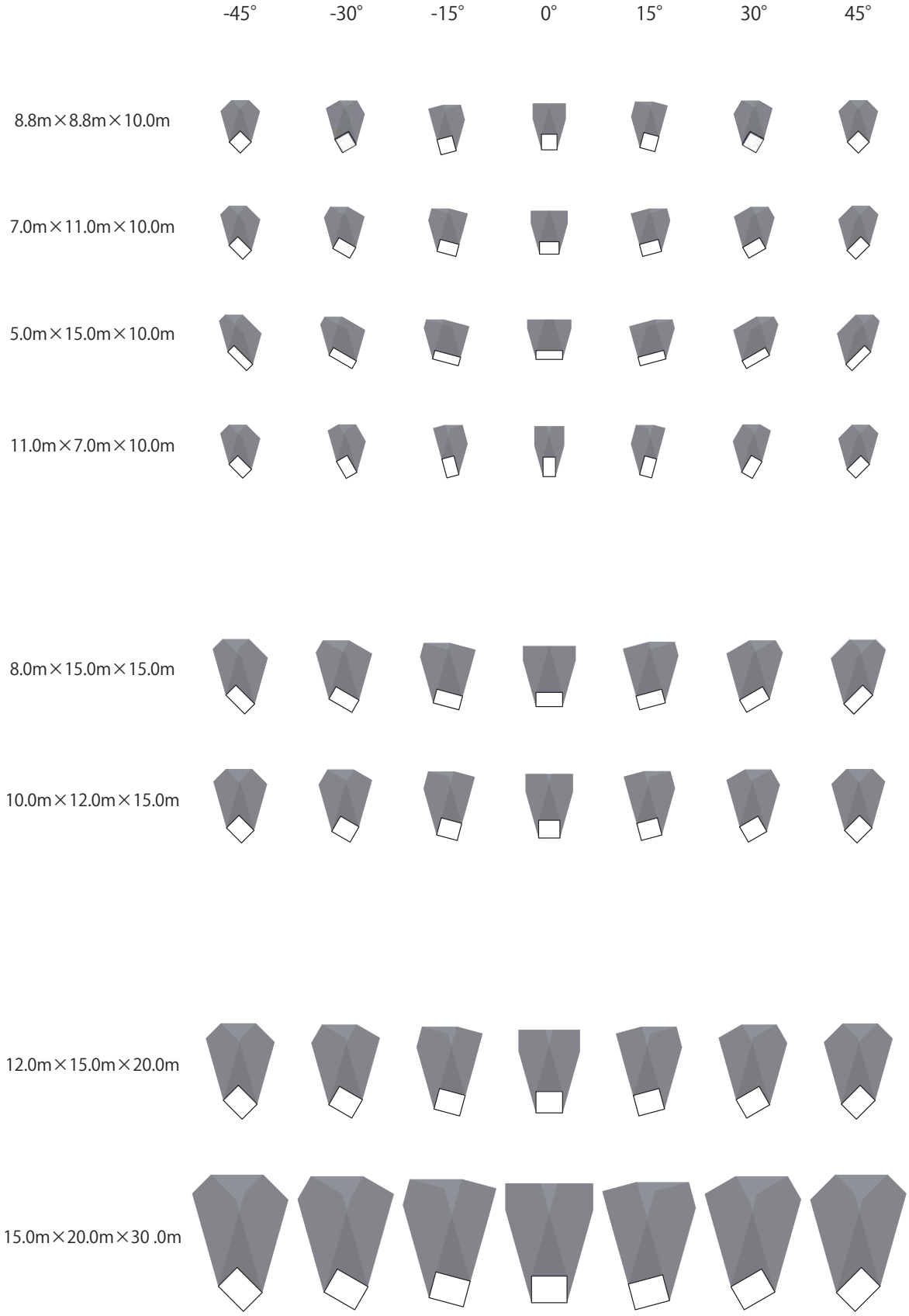
次に「デザイン・アルゴリズム」についてである。「デザイン・アルゴリズム」は人間が部分のデザインを行う際の制限条件になる。この制限はコンピュータの支援によって、人のデザインに対して高速に自動的に行われる。また、「デザイン・アルゴリズム」によってコンピュータが高速で生成した「デザイン・パターン」は、「デザイン・パラメータ」と「デザイン・アルゴリズム」が適切かどうかを判断するために使用される。

建築は敷地に 1 つづ配置されていく。建築が配置される際は、配置位置が既存建築から最小距離以上、最大距離以内であるかどうかで判断される。また配置される建築は、既存建築と重ならないこと、他の建築の排他的領域に重なっていないこと、自身の排他的領域が既存建築に重なっていないことを条件として、その位置と向きが決定される。また排他的領域同士は重なってもよいものとする

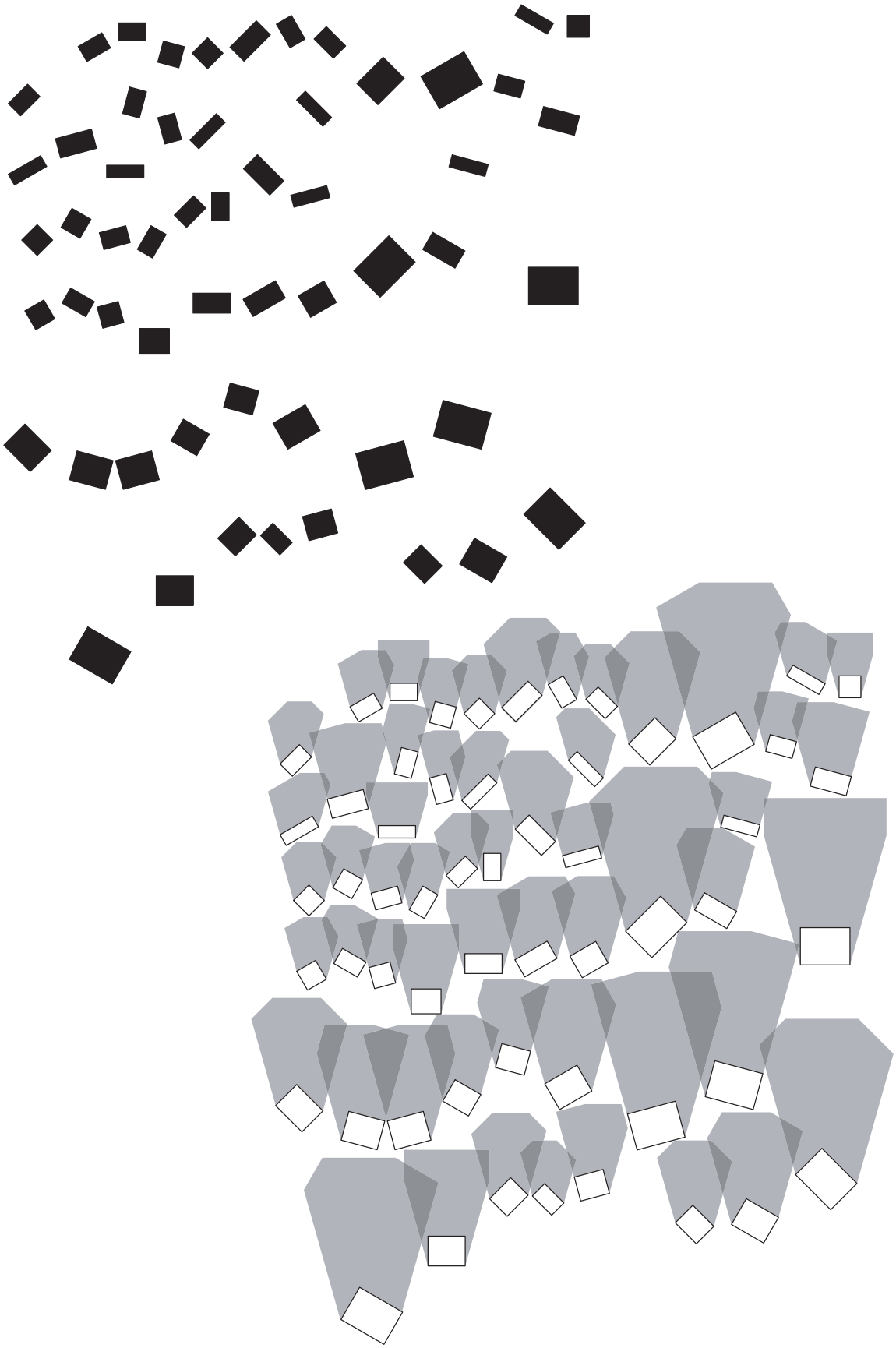
「デザイン・パターン」は 2 通の方法で作成される。1 つ目は人の手によって部分的に生成される「デザイン・パターン」である。これは「デザイン・パラメータ」と「デザイン・アルゴリズム」が人が部分をデザインする上での制限になりすぎていないかによって評価される。もう 1 つは、コンピュータによって生成される全体の「デザイン・パターン」である。ここでは、「デザイン・パラメータ」と「デザイン・アルゴリズム」が展開されていくことで生成された全体の「デザイン・パターン」が、住宅街としての機能を実現しているかが確認される。それは、全ての住宅にアクセスが可能なように道路が配置できるかという点、住宅の密度が適切なものであるという点、「非均質、非偏在」の状態であるかどうかによって判断される。これらが適切となるように、適当な「デザイン・パラメータ」と「デザイン・アル

ゴリズム」が模索される。また適応性の検証として、全体の「デザイン・パターン」は敷地形状が途中で変化する場合、異なる敷地形状に適応した場合でも生成され評価がされる。

また最終的に、デザイン解となる「デザイン・パターン」は、決定された「デザイン・パラメータ」と「デザイン・アルゴリズム」にしたがって、人の手によって生成される。



今回提案された手法で、人とコンピュータが生成可能な「デザイン・パターン」の　一例



3.6. 実験的デザインについての総括

残念ながら、ここで提案されたデザイン手法に対し具体的な検証を行うことができなかった。これについては早急な検証を行う必要があるが、それでも提案までの過程で新しい知見を得ることができた。

ここではアルゴリズムを、人によるデザイン課題の解決とコンピュータによるデザイン課題の解決を繋ぐもの設定するという新しいモデルが提案されている。デザイン課題にある 2 つの条件が存在し、それぞれが人とコンピュータに適した課題である場合、通常はそのデザイン課題を 2 つに分割することでそれを解決する。しかしアルゴリズミックな手法によるデザインでは、アルゴリズムがその 2 つを繋ぐものとして機能し、デザインプロセスを 1 つに統合するとともに、両者の欠点を補い合うようなデザインが可能になると考えられる。

4. 結論

4.1. 総括

4.1.1. アルゴリズミックなデザインを優位に活用する

アルゴリズミックな手法は、複雑だが有限時間で解決する必要のあるデザイン課題に対して用いられるべきである。

アルゴリズミックな手法の最大の利点は、人とコンピュータが協働できるという点にある。コンピュータは人と比べて圧倒的な計算処理能力を有するため、人が処理できないような複雑な問題を解いたり、その解決にかかる時間を大幅に圧縮することが可能である。こうした能力を持つコンピュータと協働できるというアルゴリズミックな手法の利点を最大限活用するためには、人が通常のデザイン手法では解決できないような複雑な問題や、限られた時間内で答えを出さなければならないようなデザイン課題に対して適応されるべきである。

アルゴリズミックな手法は、具体的で明確なデザイン課題に対して用いられるべきである。

アルゴリズミックな手法とは、デザインを問題解決のプロセスとして捉えて、その解を求めるという手法である。また、コンピュータとの協働を前提に考える上で、アルゴリズミックな手法の対象とすべきデザイン課題は、前提となる条件が具体的に定まっていることや、デザイン課題にとって解決しなければならない問題が明確に定義されていること、問題の解決手法が「デザイン・アルゴリズム」として記述できることが必要である。こうしたデザイン課題に対し、アルゴリズミックデザインはその効果を最大限発揮できる。

アルゴリズミックな手法は、デザイン解の要求される条件に感性的な許容範囲があるようなデザイン課題に用いられるべきである

アルゴリズミックな手法では、強すぎる感性的な要求が存在するデザイン解を求めることができない。アルゴリズミックな手法とは、許容範囲の大きな感性的要求の中で、それが具体化された形であるデザイン解を「デザイン・アルゴリズム」によって模索という手法だからである。

アルゴリズミックな手法の具体的な形は、「デザイン・パラメータ」、「デザイン・アルゴリズム」、「デザイン・パターン」という 3 つの視点から構築されるべきである

「デザイン・パラメータ」「デザイン・アルゴリズム」「デザイン・パターン」は、アルゴリズミックな手法によるデザインにおいて、その具体的な手法を検討する際に、着目すべき 3 つの視点である。

「デザイン・パラメータ」は、アルゴリズミックな手法において、「デザイン・アルゴリズム」によって扱われる要素を規定したり、「デザイン・アルゴリズム」が実行されるロジック上の空間と実空間をつなぐ部分である。デザイン課題と適合するように設定され、「デザイン・アルゴリズム」共にデザイン課題を適切に解決できるかに関係する。またアルゴリズミックなデザインにおいて、人が直接的にコントロール可能な要素であり、容易に調整が可能なようなものとして準備されるべきである。

「デザイン・アルゴリズム」は、デザイン課題を解決しつつ、デザイン解となるような「デザイン・パターン」を生成していく部分である。この部分は、アルゴリズミッコンピュータによって実行されるメリットが大きく、「デザイン・アルゴリズム」はデ

ザイン課題から抽出された問題を解決するようなものとして設定される。

純粋にコンピュータにのみによって実行された「デザイン・アルゴリズム」からだけでは、よいデザイン解である「デザイン・パターン」を生成することは難しい。これは、「デザイン・パターン」はあくまで、「デザイン・パラメータ」と「デザイン・アルゴリズム」によって設定されたデザイン課題を解決したものでしかないからだ。

「デザイン・パターン」の決定には、人の感性による判断が介入できる必要がある

「デザイン・パラメータ」と「デザイン・アルゴリズム」から生成される「デザイン・パターン」は、デザイン課題を十分に解決していたとしても、必ずしもよいデザイン解となるとは限らない。よいデザイン解となる「デザイン・パターン」を、必ず生成するような「デザイン・パラメータ」と「デザイン・アルゴリズム」を模索することは可能であるが、それは多くの場合徒労に終わり、最悪の場合デザイン課題を十分に解決するという「デザイン・パラメータ」と「デザイン・アルゴリズム」本来の役割を阻害することにもなる。

そのため、良いデザイン解を得るためには、生成された「デザイン・パターン」を人の感性による判断で選別するプロセスが必要不可欠であり、「デザイン・パターン」にはそうした余地があることが必要となる。

「デザイン・パターン」は評価可能な形で提示される必要がある

良いデザイン解を得るためには、「デザイン・パターン」の決定に人の感性による判断が必要になる。そのため「デザイン・パターン」は、人が評価可能な形で提示される必要がある。

しかしアルゴリズミックな手法では、評価困難である複雑な「デザイン・パターン」を扱うことが多いため、「デザイン・パターン」評価可能にするため以下のような工夫が必要である。

1. 人が「デザイン・パターン」の何をどう評価すればよいかが、明確になっている。
2. 「デザイン・パターン」が人に評価可能な複雑さに制限されて提示される。
3. 人による「デザイン・パターン」の評価が、厳密でなくてもよいような仕組みを用意する。

「デザイン・パターン」の生成はインタラクティブである必要がある

アルゴリズミックな手法では、デザイン解となる「デザイン・パターン」を「デザイン・パラメータ」によって間接的な操作する。そのため、「デザイン・パラメータ」の操作と「デザイン・パターン」の生成が人にとってインタラクションを与えるものでないと、人はアルゴリズミックな手法の中で、デザインを進展させていくことがができないのである。

「デザイン・パターン」の生成がインタラクティブであることに必要とされるのは、「デザイン・パターン」が生成される速度と、「デザイン・パラメータ」の操作性である。

「デザイン・アルゴリズム」はシンプルであるほうが良い

「デザイン・アルゴリズム」のシンプルさは「デザイン・パターン」の生成速度に影響し、それは人が「デザイン・パターン」からインタラクションを得られるかどうかに関係する。「デザイン・アルゴリズム」が十分にシンプルであれば、デザイン解となる「デザイン・パターン」が決定されるプロセスに " 他者 " が参加することが可能になる。

「デザイン・アルゴリズム」には一定の自由度があったほうが良い

「デザイン・アルゴリズム」は、デザイン課題を十分に解決できるという上で、一定の自由度が存在するほうが良い。それは、「デザイン・アルゴリズム」における選択肢の自由度であり、拡張の自由度である。

一定の自由度のある「デザイン・アルゴリズム」からは、様々なバリエーションをもった「デザイン・パターン」が生成される。「デザイン・パターン」のバリエーションが豊富であることは、人の感性によって優れているされるデザイン解が生成される可能性、人にとって予測外であるが有用で啓示的であるというデザイン解を生成する確率を上昇させる。

また、一定の自由度がある「デザイン・アルゴリズム」は、「デザイン・パターン」生成の際に、イレギュラーな要素への適応する力をもたらす。これは「デザイン・パターン」の生成時間短縮や、「デザイン・パターン」生成に " 他者 " が参加することを容易にする。

「デザイン・アルゴリズム」の挙動には予測不可能な部分がある方が良い

「デザイン・アルゴリズム」はあるデザイン要求を満たすようなものとして設定されるため、本来その挙動はしっかりと把握されている。しかし、扱われる要素の多さであったり、複数の「デザイン・アルゴリズム」が重ね合わさることで、最終的な挙動が予測不可能となることがある。予測不可能な挙動をする「デザイン・アルゴリズム」は、人が予測していなかったような「デザイン・パターン」を生成する場合がある。こうした人の予測を上回る「デザイン・アルゴリズム」は、多くの場合失敗やエラーによる人に不利益をもたらすが、時に有用で啓示的な「デザイン・パターン」を生成する場合もある。こうした可能性を残すため、「デザイン・アルゴリズム」の挙動は予測不可能な部分を持つことが望ましい。

4.1.2. アルゴリズミックデザインの優位性

人とコンピュータが協働できるデザイン手法である

既に何度も述べていることではあるが、アルゴリズミックな手法によるデザインの最大のポイントは、人とコンピュータが協働できるということである。人は空間の質や美しさといった定量化できないものを生み出すことができる一方、大量の要素や複雑な関係性を含む問題に有限時間内で解を求めることを苦手とする。コンピュータはその圧倒的な計算処理能力によって、大量の要素や複雑な関係性を含む問題において解を求めることが可能であり、デザイン課題を解決する速度も高速である。こうした異なる利点を持つ人とコンピュータが、デザインという同じ課題に対して協働で取り組むことを可能とするのが、アルゴリズミックな手法なのである。そしてこれは、後に述べるアルゴリズミックな手法によるデザインの優位性の多くに関係してくる。

デザインの速度を向上させることができる

アルゴリズミックな手法によって、人とコンピュータが協働することのメリットの 1 つが、コンピュータの高い計算処理能力をデザイン課題の解決に利用できるということである。コンピュータはその圧倒的な計算処理能力によって、高速に問題を処理することができる。コンピュータによってデザイン課題の解決速度をあげることは、デザインそのものの速度を加速させることができる。

デザイン解の探索範囲を拡大することができる

実際のデザインでは、ほぼ全てのデザイン課題において有限時間内でデザイン解を求めなければならないという制約が存在する。そのため、どれだけの速度でデザインすることができるかということは、デザイン解をどれだけ探索できるかということに直結してくる。高速にデザインすることが可能であれば、それだけ広いデザイン解の可能性を探索できるということである。デザイン解の探索範囲が拡大することは、これまで提示されてこなかったような新しいデザイン解や、より有効なデザイン解を提示できる可能性をもたらす。

またコンピュータは、人とは異なる思考方法と思考の得意分野を持つ。そのため、コンピュータが探索可能なデザイン解の領域も、必然人とは異なる。こうした異なる

デザイン解の探索領域を持つもの同士が協働することも、デザイン解の探索範囲を拡大することにつながる。

大量の要素や複雑に関係性を含むデザイン課題を扱うことができる

コンピュータが大量の要素や複雑な関係性を含む問題に解を求めることができることは、既に述べた通りである。デザインにおいて人がコンピュータと協働するということは、こうした問題を含むデザイン課題にデザイン解を求めることが可能になるということである。

これは、単に新しいデザイン対象を扱うことができるようになったということだけではない。実はこれまでデザインの対象とされていたものの中にも、本当は大量の要素や複雑な関係性を含むデザイン課題は存在していた。しかしデザイナーは、こうしたデザイン課題の複雑さに目をつぶり、それを単純なデザイン課題であると置き換えて扱ってきた。C. Arexander が、*″デザイナーの頭の中の芸術家のアトリエででっちあげる″*と呼んだデザインである。こうした、実は大量の要素や複雑な関係性を含むデザイン課題に対しても、そのデザイン課題を本当に解決するデザイン解が提示できるようになる

デザインプロセスに " 他者 " の参加が可能になる

アルゴリズミックな手法によるデザインでは、人とコンピュータが協働できるだけでなく、新しい人も協働の対象にすることができる。それは、これまでデザインに関係しながら、デザインに参加することができなかったデザインにおける " 他者 " であり、こうした " 他者 " がデザインに参加することは、デザイン解をより良いものとするための、まったく新しい視点を得ることができるということである。

適応性を必要とするデザイン課題に答えることができる

アルゴリズミックデザインにおいて、デザイン解となる「デザイン・パターン」は「デザイン・アルゴリズム」の実行結果に過ぎない。その本質は、「デザイン・アルゴリズム」が良いデザイン解である「デザイン・パターン」を生成するようにデザインされてい

るということにある。

通常のデザインにおいて、デザイン解とはある時点において確定され、以後絶対的な意味を持つとされてきた。しかし、デザインの前提条件に対する情報が不十分であった場合や、その条件が変化していくようなデザイン課題の場合においては、こうしたデザイン解の絶対性が崩れてしまう。こうしたデザインの構造が、これまで適応性を必要とするデザイン課題に答えることを困難にしていた。

しかし、デザイン解が「デザイン・アルゴリズム」の実行結果に過ぎないとなってしまえば、たとえ条件が変更されたとしても、その条件で「デザイン・アルゴリズム」を実行するだけで、新しい条件に適応したデザイン解が生成できる。アルゴリズミックな手法はこうした適応性を必要とするデザイン課題に答えることのできる、これまでにないデザイン手法なのである。

プロセスをデザイン課題として扱うことができる

「デザイン・アルゴリズム」とは、デザイン解を生成するプロセスそのものでもある。つまり、「デザイン・アルゴリズム」がデザインされているということは、デザイン解がある状態にいたる、そのプロセス自体がデザインされているということでもある。これは、上の適応を必要とするデザイン課題と似ているが、微妙に違う意味合いも持つ。適応性を持つデザインとは、本来想定された完成形があり、それが条件の変化に適応し新しい完成形となるということである。

では、こうした条件の変化と新しい完成形の生成が延々と繰り返される、完成形を持たず、だが全ての段階において完成形であることを求められるような、そんな矛盾のようなデザイン課題があったとしたらどうだろうか。これは都市のような適応性を求められ、かつ完成形を持たないようなデザイン対象において、実際に発生しうるデザイン課題である。

プロセスをデザインするアルゴリズミックな手法は、こうしたデザイン課題に対してもデザイン対象として扱うことができるのである。

4.2 今後の展開

近年、アルゴリズミックな手法によるデザインに対する認識が急激に高まっている。

現在主流の 3D モデリングソフトウェアの 1 つである Robert McNeel & Associates の Rhinoceros は、Grasshopper というプラグインを導入することで、既存のモデリング機能を大きく拡張することができる。Grasshopper は、Rhinoceros 上で自らアルゴリズムを設計し、パラメータの操作によって 3D モデルを作成できるプラグインであり、その最大の特徴は GAE(Graphical Algorithm Editor) であるということである。GAE は、プログラミングスキルのような高度な専門技術を持たないユーザーでも、グラフィカルなインターフェイスによって、アルゴリズムの設計が可能となる機能である。このように、近年はアルゴリズムな手法によってコンピュータと人が協働するようなデザインの敷居がどんどん低くなり、そのユーザーは今後ますます増加していくと考えられる。

またアルゴリズミックな手法を研究する研究者も増え、2011 年には著者も参加した、AGODE - International Symposium on Algorithmic Design for Architecture and Urban Design という日本建築学会主催のアルゴリズミックな手法によるデザインに関する国際シンポジウムも開催された。新しいアルゴリズミックな手法が次々に提案され、それらが社会に対して有用な新しいデザイン解を生成している。

このような流れの中にあって、基本に立ち戻り、アルゴリズミックな手法の優位性を改めて考察し直すという本研究は、アルゴリズミックな手法の更なる発展と、それによるデザインの社会貢献に、微力ながら一助となるのではと考えている。

文献目録

引用文献

クリストファー・アレキザンダ , 稲葉武司訳 : *"形の合成に関するノート "*, 鹿島出版会 (1978)
コスタス・テルジディス , 田中浩也 監訳, 荒岡紀子・重村珠穂・松川昌平訳 : *"アルゴリズミック・アーキテクチャ "*, 彰国社 (2010)
住宅生産振興財産編 , : *"中井検裕監修住まいのまちなみを創る　工夫された住宅地・設計事例集 "*, 建築資料研究社 (2010)

参考文献

日本建築学会編 : *" アルゴリズミック・デザインー建築・都市の新しい設計手法 "* 鹿島出版会 (2009)
柄沢祐輔 , 田中浩也 , 藤村龍至 , ドミニク・チェン , 松川昌平 , メディア・デザイン研究所編 : *" 設計の設計 "* 稲葉武司訳 INAX 出版 (2011)
ノイズアーキテクツ : *"Rhinoceros + Grasshopper 建築デザイン実践ハンドブック (建築文化シナジー)"*, 彰国社 (2011)
渡辺誠 : *" 建築は、柔らかい科学に近づくー INDUCTION DESIGN / 進化設計論 "* , 建築資料研

究所 (2002)
浅野哲夫 : *"計算幾何―理論の基礎から実装まで (アルゴリズム・サイエンスシリーズ―数理技法編)"* , 共立出版 (2007)

参考論文

阿部祐一 , 池田靖史 : *"空間性を手がかりとしたコンピュータシミュレーションによる自己組織的、プロセス的手法を用いた住宅街の配置デザインに関する研究 "* , 第 34 回情報・システム・利用・技術シンポジウム論文集 (2011)

謝辞

長かった僕の大学生活も、この修士論文を終えたことでようやく 1 区切りとなりそうです。学生としても、人間としてもいたらない点ばかりであった僕が、この研究を、そして大学生活を全うできたのは、ひとえに人への僕を支えてくださった皆様のおかげです。

池田靖史先生

修論の主査をお引き受けいただきありがとうございました。

先生からご指導いただいた 3 年間は、僕の大学・大学院生活 6 年間の半分に相当します。期間としては長いようで短い期間でしたが、先生のご指導から得られたものを考えると、驚くほど短い期間でした。

田中浩也先生、中島直人先生

副査をお引きいただいきありがとうございました。

僕が至らぬせいで、あまりご指導をいただけなかったことが悔やまれます。少ない機会にも関わらず貴重なご意見をいただけました。

小林博人先生、松原弘典先生、松川昌平先生

コンピュータへの興味から SFC にきた僕が、建築に興味を持つきっかけになったのが小林先生の指導でした。

また、建築に興味以上の熱意をもてなかった僕が、建築との関係を考え直すきっかけとなったのは、松原先生の熱心なご指導でした。

そして建築とコンピュータの間で揺れ続けていた自身の興味に、一条の光明を見出すことができたのは松川先生の授業がきっかけでした。

池田靖史研究室の青山さん、谷垣内くん、キムくん

大学院生活では、研究、プライベートともに、たくさん迷惑をかけ、たくさん助け助てもらいました。僕が大学院で研究を続けられたのはみんなのおかげです。

両親、妹

僕の長い大学生活に、気長につきあってくれてありがとう。

そして、僕に関わったくれた全ての方に感謝を。。。

付録

ソースコード

デザイン事例 1 : Alternative Suburban

```
import processing.dxf.*;
import processing.pdf.*;
import processing.opengl.*;

ArrayList Houses;
ArrayList UniSpaces, Lines, Neighbors;
int HperY;
int fileNum;
int[] lifeWidth;
int[][] grid;
int count;
int dispMode;
float worldLen, worldLenX, worldLenY, scaleDis;
float areaSizeX, areaSizeY;
float marginX, marginY;
float rctSize, rctSizeX, rctSizeY, shapeBtw;
float gdnSize, gdnSizeX, gdnSizeY, gdnAng, gdnShapeVar;
float roadWidth;
float[][] Area, t_Area;
int houseID;
int[] HouseIDs, revHouseIDs;

int houseInfoMode;

PFont font16, font12, font9, font6;
int moviefps;

String message1, message2, message3, message4;

int t;
boolean export, exportFlg, goRunFlg, frameAdvFlg, houseSelectFlg, addManual, addedHouse, houseDragged, pressButton;

PrintWriter areaRate, gdnType, sunshine, vArea;
PrintWriter[] appVoid;
String[] witerName;
Camera cam;
Axis axis;
PGraphics pg3D, pg2D, pgFP, pgText, pdf;

float pressMouseX, pressMouseY, dragAng;

void setup(){
  moviefps=5;
  size(600, 500);
  background(220);
  //frameRate(10);

  export=true;

  initSetting();
  ControlP5Setup();

  culcThread generator = new culcThread();
  Thread genThread = new Thread(generator);
  genThread.start();
}

void draw(){
  background(220);
  Display3D();
  Display(pg2D, 5, false, false);
  textDisplay();
  if(dispMode==0){
    image(pg2D, 0, 0);
    image(pgText, 0, height-100);
  }else if(dispMode==1){
    image(pg3D, 0, 0);
  }else if(dispMode==2){
    image(pg2D, width-200, height-200, 200, 200);
    image(pg3D, 0, 0);
    image(pgText, 0, height-100);
  }else{
    image(pg2D, 0, 0);
    image(pg3D, width-200, height-200, 200, 200);
    image(pgText, 0, height-100);
  }

  fill(200);
  noStroke();
  rect(width-200, 0, 200, height-200);
  stroke(0);
  strokeWeight(1);
  line(width-200, 0, width-200, height);
  line(width-200, height-200, width, height-200);

  if(!goRunFlg){
    if(addManual){
      if(houseDragged){
        if(!pressButton){
          noFill();
          stroke(255, 0, 0);
          dragAng=vectorAngle(new float[][]{{pressMouseX+marginX, pressMouseY+marginY}, {mouseX, mouseY}})+PI/2;
          line(pressMouseX+marginX, pressMouseY+marginY, mouseX, mouseY);
          fill(0, 100, 255, 220);
          stroke(0, 0, 255);
          drawShape(makeSquare(new float[][]{{pressMouseX, pressMouseY}, {rctSizeX, rctSizeY}, {dragAng}}));
          fill(0, 60, 60, 100);
          stroke(0, 255, 0, 100);
```

```
        strokeWeight(0.5);
          drawShape(makeTrapezoid2(new float[][]{movePoint(new float[]{pressMouseX, pressMouseY}, (rctSizeY+gdnSizeY)/2, dragAng),{gdnSizeX,
gdnSizeY}, {dragAng, 0}, {gdnAng}}));
      }
    }else{
      if(addedHouse){
        fill(0, 100, 255, 220);
        stroke(0, 0, 255);
        drawShape(makeSquare(new float[][]{{pressMouseX, pressMouseY}, {rctSizeX, rctSizeY}, {dragAng}}));
        fill(0, 60, 60, 100);
        stroke(0, 255, 0, 100);
        strokeWeight(0.5);
          drawShape(makeTrapezoid2(new float[][]{movePoint(new float[]{pressMouseX, pressMouseY}, (rctSizeY+gdnSizeY)/2, dragAng),{gdnSizeX,
gdnSizeY}, {dragAng, 0}, {gdnAng}}));
      }else{
        if((mouseX<width-200)&&(mouseY<height-100)){
          fill(0, 100, 255, 220);
          stroke(0, 0, 255);
          rect(mouseX-rctSizeX/2, mouseY-rctSizeY/2, rctSizeX, rctSizeY);
        }
      }
    }
  }else{

    int nearHouseID=getNearHouse(new float[]{mouseX, mouseY}, 30);
    if(nearHouseID!=-1){
      noFill();
      stroke(255, 0, 0);
      strokeWeight(0.5);
      House t_House=(House)Houses.get(nearHouseID);
      drawShape(t_House.makeShape());
      houseSelectFlg=true;
      fill(255);
      textFont(font9);
      //text(t_House.id, t_House.posX, t_House.posY);
      switch(houseInfoMode){
        case 1:
          stroke(255, 255, 100, 150);
          strokeWeight(0.5);
          noFill();
          float[][][] AppLine1=t_House.getAppLine(100, 0);
          if(AppLine1.length>0){
            for(int i=0; i<AppLine1.length; i++){
              drawShape(AppLine1[i]);
            }
          }
          break;
        case 2:
          stroke(255, 255, 100, 150);
          strokeWeight(0.5);
          noFill();
          float[][][] AppLine2=t_House.getAppLine(100, 1);
          if(AppLine2.length>0){
            for(int i=0; i<AppLine2.length; i++){
              drawShape(AppLine2[i]);
            }
          }
          break;
        case 3:
          stroke(255, 255, 100, 150);
          strokeWeight(0.5);
          noFill();
          float[][][] AppLine3=t_House.getAppLine(100, 2);
          if(AppLine3.length>0){
            for(int i=0; i<AppLine3.length; i++){
              drawShape(AppLine3[i]);
            }
          }
          break;
        case 4:
          stroke(255, 255, 100, 150);
          strokeWeight(0.5);
          noFill();
          float[][][] AppLine4=t_House.getAppLine(100, 3);
          if(AppLine4.length>0){
            for(int i=0; i<AppLine4.length; i++){
              drawShape(AppLine4[i]);
            }
          }
          break;
        case 5:
          stroke(255, 255, 100, 150);
          strokeWeight(2);
          noFill();
          float[][][] AppLine5=t_House.getAppLine(100, 4);
          if(AppLine5.length>0){
            for(int i=0; i<AppLine5.length; i++){
              drawShape(AppLine5[i]);
            }
          }
          println();
          break;
        case 6:
          println("houseInfoMode : "+houseInfoMode);
          noStroke();
          fill(255, 150, 0, 80);
          drawShape(t_House.getVoronoi());
          break;
```

```
        case 7:
          println("houseInfoMode : "+houseInfoMode);
          stroke(0, 0, 255, 150);
          strokeWeight(0.5);
          noFill();
          float[][][] ClosestConnections=t_House.getClosestConnection();
          if(ClosestConnections.length>0){
            for(int i=0; i<ClosestConnections.length; i++){
              if(ClosestConnections[i]!=null){
                drawShape(ClosestConnections[i]);
              }
            }
          }
          break;
        case 8:
          println("houseInfoMode : "+houseInfoMode);
          stroke(0, 255, 0, 150);
          strokeWeight(0.5);
          noFill();
          float[][][] NeighborConnections=t_House.getNeighborConnection();
          if(NeighborConnections.length>0){
            for(int i=0; i<NeighborConnections.length; i++){
              drawShape(NeighborConnections[i]);
            }
          }
          break;
      }
    }else{
      houseSelectFlg=false;
    }
  }
 }
}

class Axis{
 float len;
 float[] P=new float[3];
 float pit=10.;
 Axis(float _len){
  len=_len;
  P=new float[]{0, 0, 0};
 }

 void len(float _len){
  len=_len;
 }

 void shiftPoint(float[] _P){
  P=_P;
 }

 void Display(){
  /*
  for(int i=0;i<(len/pit);i++){
    pg3D.stroke(255, 255, 255, 100);
      pg3D.strokeWeight(0.5);
    pg3D.line(P[0]+i*pit, P[1], P[2], P[0]+i*pit, P[1]+pit*int(len/pit), P[2]);
    pg3D.line(P[0], P[1]+i*pit, P[2], P[0]+pit*int(len/pit), P[1]+i*pit, P[2]);
  }
  */
  pg3D.stroke(255,0,0);
  pg3D.strokeWeight(0.5);
  pg3D.line(P[0], P[1], P[2], P[0]+len, P[1], P[2]);
  pg3D.stroke(0,255,0);
  pg3D.strokeWeight(0.5);
  pg3D.line(P[0], P[1], P[2], P[0], P[1]+len, P[2]);
  pg3D.stroke(0,0,255);
  pg3D.strokeWeight(0.5);
  pg3D.line(P[0], P[1], P[2], P[0], P[1], P[2]+len);
 }
}

int[] deleteFromIndex(int[] _Arr, int _index){
 if((_Arr.length>0)&&(_Arr.length>_index)){
  if(_index==0){
    _Arr=(int[])subset(_Arr, 1);
  }else if(_index>=(_Arr.length-1)){
    _Arr=(int[])shorten(_Arr);
  }else{
    _Arr=(int[])concat((int[])subset(_Arr, 0, _index), (int[])subset(_Arr, _index+1));
  }
 }
 return _Arr;
}

float[] deleteFromIndex(float[] _Arr, int _index){
 if((_Arr.length>0)&&(_index<_Arr.length)){
  if(_index==0){
    _Arr=(float[])subset(_Arr, 1);
  }else if(_index>=(_Arr.length-1)){
    _Arr=(float[])shorten(_Arr);
  }else{
    _Arr=(float[])concat((float[])subset(_Arr, 0, _index), (float[])subset(_Arr, _index+1));
  }
 }
 return _Arr;
}
```

```
int[] deleteFromElement(int[] _Arr, int _element){
  int[] newArr=new int[0];
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length; i++){
      if(_Arr[i]!=_element){
        newArr=(int[])append(newArr, _Arr[i]);
      }
    }
  }
  return newArr;
}

float[] deleteFromElement(float[] _Arr, float _element){
  float[] newArr=new float[0];
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length; i++){
      if(_Arr[i]!=_element){
        newArr=(float[])append(newArr, _Arr[i]);
      }
    }
  }
  return newArr;
}

int[] deleteOverlap(int[] _Arr){
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length-1; i++){
      for(int j=i+1; j<_Arr.length; j++){
        if(_Arr[i]==_Arr[j]){
          _Arr[j]=-1;
        }
      }
    }
  }
  _Arr=deleteFromElement(_Arr, -1);

  return _Arr;
}

float[] deleteOverlap(float[] _Arr){
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length-1; i++){
      for(int j=i+1; j<_Arr.length; j++){
        if(_Arr[i]==_Arr[j]){
          _Arr[j]=-1;
        }
      }
    }
  }
  _Arr=deleteFromElement(_Arr, -1);

  return _Arr;
}

int getIndex(int[] _Arr, int _element){
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length; i++){
      if(_Arr[i]==_element){
        return i;
      }
    }
  }
  return -1;
}

int getIndex(float[] _Arr, float _element){
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length; i++){
      if(_Arr[i]==_element){
        return i;
      }
    }
  }
  return -1;
}

// 角度
float vectorAngle(float[] Arr){
  float _angle=acos(Arr[0]/sqrt(sq(Arr[0])+sq(Arr[1])));
  if(Arr[1]<0){
    return PI*2-_angle;
  }else{
    return _angle;
  }
}

// 角度
float vectorAngle(float[] Arr1, float[] Arr2){
  return vectorAngle(new float[]{Arr2[0]-Arr1[0], Arr2[1]-Arr1[1]});
}

// 角度
float vectorAngle(float[][] Arr){
  return vectorAngle(new float[]{Arr[1][0]-Arr[0][0], Arr[1][1]-Arr[0][1]});
}

// 図形の中心の座標取得
float[] shapeCenter(float[][] Arr){
```

```
  float sum_x=0;
  float sum_y=0;
  for(int i=0; i<Arr.length; i++){
    sum_x=sum_x+Arr[i][0];
    sum_y=sum_y+Arr[i][1];
  }
  return new float[]{sum_x/Arr.length, sum_y/Arr.length};
}

// 図形の大まかな大きさ取得
float shapeSize(float[][] Arr){
  float[] Shape_c=shapeCenter(Arr);
  float Shape_s=0;
  for(int i=0; i<Arr.length; i++){
    if(Shape_s<distPtoP(Arr[i], Shape_c))Shape_s=distPtoP(Arr[i], Shape_c);
  }
  return Shape_s;
}

// 図形の面積取得
float shapeArea(float[][] Arr){
  float shparea=0;
  for(int i=0; i<Arr.length; i++){
    shparea=shparea+(Arr[i%Arr.length][0]-Arr[(i+1)%Arr.length][0])*(Arr[i%Arr.length][1]+Arr[(i+1)%Arr.length][1])/2;
  }
  return abs(shparea);
}

// 図形同士の交差面積取得
float crrShapeArea(float[][] Arr1, float[][] Arr2){
  ArrayList Points;
  int count=0;
  Points=new ArrayList();
  for(int i=0; i<Arr1.length; i++){
    for(int j=0; j<Arr2.length; j++){
      float dx1=Arr1[i][0]-Arr1[(i+1)%Arr1.length][0];
      float dy1=Arr1[i][1]-Arr1[(i+1)%Arr1.length][1];
      float dx2=Arr2[j][0]-Arr2[(j+1)%Arr2.length][0];
      float dy2=Arr2[j][1]-Arr2[(j+1)%Arr2.length][1];
      if(crrSegment(new float[][]{Arr1[i], Arr1[(i+1)%Arr1.length]}, new float[][]{Arr2[j], Arr2[(j+1)%Arr2.length]})){
        float A=dy1/dx1;
        float B=dy2/dx2;
        float x=((Arr1[(i+1)%Arr1.length][0]*A)-Arr1[(i+1)%Arr1.length][1]-(Arr2[(j+1)%Arr2.length][0]*B)+Arr2[(j+1)%Arr2.length][1])/(A-B);
        float y=A*(x-Arr1[(i+1)%Arr1.length][0])+Arr1[(i+1)%Arr1.length][1];
        Points.add(new float[]{x,y});
        count++;
      }
      if(InShape(Arr1[(i+1)%Arr1.length], Arr2)){
        Points.add(Arr1[(i+1)%Arr1.length]);
      }
      if(count>1)break;
    }
    if(count>1)break;
  }
  float[][] crr=new float[Points.size()][2];
  for(int i=0; i<Points.size(); i++){
    crr[i]=(float[])Points.get(i);
  }
  return shapeArea(crr);
}

// 原点と座標間の距離取得
float distPtoP(float[] Arr){
  return dist(Arr[0], Arr[1], 0, 0);
}

// 座標間の距離取得
float distPtoP(float[] Arr1, float[] Arr2){
  return dist(Arr1[0], Arr1[1], Arr2[0], Arr2[1]);
}

// 座標間の距離取得
float distPtoP(float[][] Arr){
  return dist(Arr[0][0], Arr[0][1], Arr[1][0], Arr[1][1]);
}

// 線分と座標の距離取得
float distLtoP(float[][] Arr1, float[] Arr2){
  float dx=Arr1[1][0]-Arr1[0][0];
  float dy=Arr1[1][1]-Arr1[0][1];
  float a=dx*dx+dy*dy;
  if(a==0)return dist(Arr1[0][0] ,Arr1[0][1], Arr2[0], Arr2[1]);
  float b=dx*(Arr1[0][0]-Arr2[0])+dy*(Arr1[0][1]-Arr2[1]);
  float t=-(b/a);
  if(t<0)t=0;
  if(t>1)t=1;
  float x=t*dx+Arr1[0][0];
  float y=t*dy+Arr1[0][1];
  return dist(x, y, Arr2[0], Arr2[1]);
}

// 座標間の中心座標取得
float[] centerPtoP(float[] Arr1, float[] Arr2){
  return new float[]{Arr2[0]+(Arr1[0]-Arr2[0])/2, Arr2[1]+(Arr1[1]-Arr2[1])/2};
}

// 線分と座標の最短距離の点生成
float[] minLtoP(float[][] Arr1, float[] Arr2){
```

```
  float dx=Arr1[1][0]-Arr1[0][0];
  float dy=Arr1[1][1]-Arr1[0][1];
  float a=dx*dx+dy*dy;
  if(a==0)return Arr1[0];
  float b=dx*(Arr1[0][0]-Arr2[0])+dy*(Arr1[0][1]-Arr2[1]);
  float t=-(b/a);
  if(t<0)t=0;
  if(t>1)t=1;
  float x=t*dx+Arr1[0][0];
  float y=t*dy+Arr1[0][1];
  return new float[]{x, y};
}

// 線分と座標の最短距離の点から、ある角度分振れた場所にある点の生成
float[][] AminLtoP(float[][] Arr1, float[] Arr2, float[] Arr3, float _ang){
  float dx=Arr2[0]-Arr3[0];
  float dy=Arr2[1]-Arr3[1];
  float distPtoL=sqrt(dx*dx+dy*dy);
  float[] _point1=new float[]{Arr3[0]+dx*tan(_ang), Arr3[1]+dy*tan(_ang)};
  float[] _point2=new float[]{Arr3[0]-dx*tan(_ang), Arr3[1]-dy*tan(_ang)};
  if((distLtoP(Arr1, _point1)==0)&&(distLtoP(Arr1, _point2)==0)){
    return new float[][]{_point1, _point2};
  }else if(distLtoP(Arr1, _point1)==0){
    return new float[][]{_point1};
  }else if(distLtoP(Arr1, _point2)==0){
    return new float[][]{_point2};
  }else{
    return new float[][]{Arr3};
  }
}

// 原点を中心とした座標の回転
float[] rotatePoint(float[] Arr, float angle){
  float[] new_p={Arr[0]*cos(angle)-Arr[1]*sin(angle), Arr[0]*sin(angle)+Arr[1]*cos(angle)};
  return new_p;
}

// 特定の座標を中心とした座標の回転
float[] rotatePoint(float[] Arr1, float[] Arr2, float angle){
  float[] new_p={Arr2[0]+(Arr1[0]-Arr2[0])*cos(angle)-(Arr1[1]-Arr2[1])*sin(angle), Arr2[1]+(Arr1[0]-Arr2[0])*sin(angle)+(Arr1[1]-Arr2[1])*cos(angle)};
  return new_p;
}

// 中心の座標を中心とした図形の回転
float[][] rotateShape(float[][] Arr, float angle){
  float[][] new_s=new float[Arr.length][2];
  float[] Shape_c=shapeCenter(Arr);
  for(int i=0; i<Arr.length; i++){
    float[] new_p=rotatePoint(Arr[i], Shape_c, angle);
    new_s[i]=new_p;
  }
  return new_s;
}

// 特定の座標を中心とした図形の回転
float[][] rotateShape(float[][] Arr1, float[] Arr2, float angle){
  float[][] new_s=new float[Arr1.length][2];
  for(int i=0; i<Arr1.length; i++){
    float[] new_p=rotatePoint(Arr1[i], Arr2, angle);
    new_s[i]=new_p;
  }
  return new_s;
}

// 座標の角度方向への移動
float[] movePoint(float[] Shape, float Length, float Direct){
  float shapeCenterX=Shape[0];
  float shapeCenterY=Shape[1];
  float cosS=cos(Direct)*Length;
  float sinS=sin(Direct)*Length;
  return new float[]{shapeCenterX+sinS, shapeCenterY-cosS};
}

// 図形の角度方向への移動 1
float[] moveShape(float[] Shape, float Length, float Direct){
  float shapeCenterX=Shape[0];
  float shapeCenterY=Shape[1];
  float cosS=cos(Direct)*Length;
  float sinS=sin(Direct)*Length;
  return new float[]{shapeCenterX+sinS, shapeCenterY-cosS, Shape[2], Shape[3]};
}

// 図形の角度方向への移動 2
float[][] moveShape(float[][] Shape, float Length, float Direct){
  float shapeCenterX=Shape[0][0];
  float shapeCenterY=Shape[0][1];
  float cosS=cos(Direct)*Length;
  float sinS=sin(Direct)*Length;
  return new float[][]{{shapeCenterX+sinS, shapeCenterY-cosS}, {Shape[1][0], Shape[1][1]},};
}

// 線分と座標の交点生成
float[] crrLtoPPoint(float[][] Arr1, float[] Arr2){
  if(crrLtoP(Arr1, Arr2))
    return Arr2;
  else
    return new float[0];
}
```

```
//2 直線の交点生成
float[] crrLinePoint(float[][] Arr1, float[][] Arr2){

  float _slopeX1=Arr1[1][0]-Arr1[0][0];
  float _slopeY1=Arr1[1][1]-Arr1[0][1];
  float _slopeX2=Arr2[1][0]-Arr2[0][0];
  float _slopeY2=Arr2[1][1]-Arr2[0][1];

  if((_slopeX1==0)&&(_slopeY1==0)&&(_slopeX2==0)&&(_slopeY2==0)){
    if(distPtoP(Arr1[0], Arr2[0])==0){
      return Arr1[0];

    }else{
      return new float[2];

    }
  }else if((abs(_slopeX1)<0.0005)&&(abs(_slopeY1)<0.0005)){
    return crrLtoPPoint(Arr2, Arr1[0]);

  }else if((abs(_slopeX2)<0.0005)&&(abs(_slopeY2)<0.0005)){
    return crrLtoPPoint(Arr1, Arr2[0]);

  }else if((abs(_slopeY1)<0.0005)&&(abs(_slopeX2)<0.0005)){
    return new float[]{Arr2[0][0], Arr1[0][1]};

  }else if((abs(_slopeX1)<0.0005)&&(abs(_slopeY2)<0.0005)){
    return new float[]{Arr1[0][0], Arr2[0][1]};

  }else if(abs(_slopeX1)<0.0005){
    float _slope2=_slopeY2/_slopeX2;
    float _crrPX=Arr1[0][0];
    float _crrPY=Arr2[0][1]+(Arr1[0][0]-Arr2[0][0])*_slope2;
    return new float[]{_crrPX, _crrPY};

  }else if(abs(_slopeX2)<0.0005){
    float _slope1=_slopeY1/_slopeX1;
    float _crrPX=Arr2[0][0];
    float _crrPY=Arr1[0][1]+(Arr2[0][0]-Arr1[0][0])*_slope1;
    return new float[]{_crrPX, _crrPY};

  }else if(abs(_slopeY1)<0.0005){
    float _slope2=_slopeX2/_slopeY2;
    float _crrPY=Arr1[0][1];
    float _crrPX=Arr2[0][0]+(Arr1[0][1]-Arr2[0][1])*_slope2;
    return new float[]{_crrPX, _crrPY};

  }else if(abs(_slopeY2)<0.0005){
    float _slope1=_slopeX1/_slopeY1;
    float _crrPY=Arr2[0][1];
    float _crrPX=Arr1[0][0]+(Arr2[0][1]-Arr1[0][1])*_slope1;
    return new float[]{_crrPX, _crrPY};

  }else if((abs(_slopeX1)>0.0005)&&(abs(_slopeX2)>0.0005)){
    float _slope1=_slopeY1/_slopeX1;
    float _slope2=_slopeY2/_slopeX2;
    if(_slope1-_slope2==0){
      return new float[0];

    }else{
      float _crrPX=(_slope1*Arr1[0][0]-_slope2*Arr2[0][0]+Arr2[0][1]-Arr1[0][1])/(_slope1-_slope2);
      float _crrPY=_slope1*(_crrPX-Arr1[0][0])+Arr1[0][1];
      return new float[]{_crrPX, _crrPY};

    }
  }else{
    float _slope1=_slopeX1/_slopeY1;
    float _slope2=_slopeX2/_slopeY2;
    if(_slope1-_slope2==0){
      return new float[0];

    }else{
      float _crrPY=(_slope1*Arr1[0][1]-_slope2*Arr2[0][1]+Arr2[0][0]-Arr1[0][0])/(_slope1-_slope2);
      float _crrPX=_slope1*(_crrPY-Arr1[0][1])+Arr1[0][0];
      return new float[]{_crrPX, _crrPY};

    }
  }
}

// 線分と線分の交点生成
float[] crrSegPoint(float[][] Arr1, float[][] Arr2){
  if(crrSegment(Arr1, Arr2)){
    return crrLinePoint(Arr1, Arr2);
  }else{
    return new float[0];
  }
}

// 図形のバウンディングボックス同士の交差判定
boolean crrBBox(float[][] Arr1, float[][] Arr2){
  float dx1=Arr1[0][0];
  float dy1=Arr1[0][1];
  float dx2=Arr1[1][0];
  float dy2=Arr1[1][1];
  float dx3=Arr2[0][0];
  float dy3=Arr2[0][1];
  float dx4=Arr2[1][0];
```

```
  float dy4=Arr2[1][1];

  if(dx1>=dx2){
    if(((dx1<dx3)&&(dx1<dx4))||((dx2>dx3)&&(dx2>dx4)))
      return false;
  }else{
    if(((dx2<dx3)&&(dx2<dx4))||((dx1>dx3)&&(dx1>dx4)))
      return false;
  }

  if(dy1>=dy2){
    if(((dy1<dy3)&&(dy1<dy4))||((dy2>dy3)&&(dy2>dy4)))
      return false;
  }else{
    if(((dy2<dy3)&&(dy2<dy4))||((dy1>dy3)&&(dy1>dy4)))
      return false;
  }

  return true;
}

// 線分と座標の交差判定
boolean crrLtoP(float[][] Arr1, float[] Arr2){
  if(distLtoP(Arr1, Arr2)<=0.005){
    return true;
  }else{
    return false;
  }
}

// 線分と線分の交差判定
boolean crrSegment(float[][] Arr1, float[][] Arr2){
  if(!crrBBox(Arr1, Arr2))return false;

  float dx1=Arr1[0][0];
  float dy1=Arr1[0][1];
  float dx2=Arr1[1][0];
  float dy2=Arr1[1][1];
  float dx3=Arr2[0][0];
  float dy3=Arr2[0][1];
  float dx4=Arr2[1][0];
  float dy4=Arr2[1][1];

  float set1=dx1-dx2;
  float set2=dy1-dy2;
  float set3=dx3-dx4;
  float set4=dy3-dy4;
  float set5=dx3-dx1;
  float set6=dy3-dy1;
  if((set1*set6-set2*set5)*
    (set1*(dy4-dy1)+set2*(dx1-dx4))>0)
      return false;
  if((-(set3*set6)+set4*set5)*
    (set3*(dy2-dy3)+set4*(dx3-dx2))>0)
      return false;
  return true;
}

// 図形と座標の包含判定
boolean InShape(float[] Arr1, float[][] Arr2){
  int count=0;
  for(int i=0; i<Arr2.length; i++) {
    float[] vector1={Arr2[i][0]-Arr1[0], Arr2[i][1]-Arr1[1]};
    float[] vector2={Arr2[(i+1)%Arr2.length][0]-Arr1[0], Arr2[(i+1)%Arr2.length][1]-Arr1[1]};
    float ext=vector1[0]*vector2[1]-vector1[1]*vector2[0];
    if((vector1[1]>=0)&&(vector2[1]<0)&&(ext>0)){
      count++;
    }else if((vector1[1]<0)&&(vector2[1]>=0)&&(ext<0)){
      count++;
    }
  }
  if((count%2)==1) {
    return true;
  }else{
    return false;
  }
}

// 図形と線分の交差判定
boolean crrShape(float[][] Arr1, float[] Arr2, float[] Arr3){
  for(int i=0; i<Arr1.length; i++){
    if(InShape(Arr2, Arr1))return true;
    if(InShape(Arr3, Arr1))return true;
    float dx1=Arr1[i][0]-Arr1[(i+1)%Arr1.length][0];
    float dy1=Arr1[i][1]-Arr1[(i+1)%Arr1.length][1];
    float dx2=Arr2[0]-Arr3[0];
    float dy2=Arr2[1]-Arr3[1];
    float dx3=Arr2[0]-Arr1[i][0];
    float dy3=Arr2[1]-Arr1[i][1];
    if(crrSegment(new float[][]{Arr1[i], Arr1[(i+1)%Arr1.length]}, new float[][]{Arr2, Arr3}))
      return true;
  }
  return false;
}

// 図形と図形の交差判定
boolean crrShape(float[][] Arr1, float[][] Arr2){
  for(int i=0; i<Arr1.length; i++){
```

```
    if(InShape(Arr1[i], Arr2))return true;
    for(int j=0; j<Arr2.length; j++){
      if(InShape(Arr2[j], Arr1))return true;
      float dx1=Arr1[i][0]-Arr1[(i+1)%Arr1.length][0];
      float dy1=Arr1[i][1]-Arr1[(i+1)%Arr1.length][1];
      float dx2=Arr2[j][0]-Arr2[(j+1)%Arr2.length][0];
      float dy2=Arr2[j][1]-Arr2[(j+1)%Arr2.length][1];
      float dx3=Arr2[j][0]-Arr1[i][0];
      float dy3=Arr2[j][1]-Arr1[i][1];
      if(crrSegment(new float[][]{Arr1[i], Arr1[(i+1)%Arr1.length]}, new float[][]{Arr2[j], Arr2[(j+1)%Arr2.length]}))
        return true;
    }
  }
  return false;
}

// 図形と図形の包含判定
boolean InShape(float[][] Arr1, float[][] Arr2){
  for(int i=0; i<Arr1.length; i++){
    if(!InShape(Arr1[i], Arr2))return false;
  }
  return true;
}

// 中心、サイズ、角度から長方形の生成
float[][] makeSquare(float[][] Shape){
  float shapeCenterX=Shape[0][0];
  float shapeCenterY=Shape[0][1];
  float shapeSizeh=Shape[1][0]/2;
  float shapeSizev=Shape[1][1]/2;
  float sinSh=sin(Shape[2][0])*shapeSizeh;
  float cosSh=cos(Shape[2][0])*shapeSizeh;
  float sinSv=sin(Shape[2][0])*shapeSizev;
  float cosSv=cos(Shape[2][0])*shapeSizev;

  return new float[][]{
    {shapeCenterX-cosSh+sinSv, shapeCenterY-sinSh-cosSv},
    {shapeCenterX+cosSh+sinSv, shapeCenterY+sinSh-cosSv},
    {shapeCenterX+cosSh-sinSv, shapeCenterY+sinSh+cosSv},
    {shapeCenterX-cosSh-sinSv, shapeCenterY-sinSh+cosSv}
  };
}

// 中心、サイズ、角度から三角形の生成
float[][] makeTriangle(float[][] Shape){
  float shapeCenterX=Shape[0][0];
  float shapeCenterY=Shape[0][1];
  float shapeSizeh=Shape[1][0]/2;
  float shapeSizev=Shape[1][1]/2;
  float sinSh=sin(Shape[2][0])*shapeSizeh;
  float cosSh=cos(Shape[2][0])*shapeSizeh;
  float sinSv=sin(Shape[2][0])*shapeSizev;
  float cosSv=cos(Shape[2][0])*shapeSizev;
  float sintanSh=sin(Shape[2][0])*(tan(Shape[3][0])*shapeSizev*2+shapeSizeh);
  float costanSh=cos(Shape[2][0])*(tan(Shape[3][0])*shapeSizev*2+shapeSizeh);
  return new float[][]{
    {shapeCenterX-costanSh+sinSv, shapeCenterY-sintanSh-cosSv},
    {shapeCenterX+costanSh+sinSv, shapeCenterY+sintanSh-cosSv},
    {shapeCenterX, shapeCenterY}
  };
}

// 中心、サイズ、角度から台形の生成
float[][] makeTrapezoid(float[][] Shape){
  float shapeCenterX=Shape[0][0];
  float shapeCenterY=Shape[0][1];
  float shapeSizeh=Shape[1][0]/2;
  float shapeSizev=Shape[1][1]/2;
  float sinSh=sin(Shape[2][0])*shapeSizeh;
  float cosSh=cos(Shape[2][0])*shapeSizeh;
  float sinSv=sin(Shape[2][0])*shapeSizev;
  float cosSv=cos(Shape[2][0])*shapeSizev;
  float sintanSh=sin(Shape[2][0])*(tan(Shape[3][0])*shapeSizev*2+shapeSizeh);
  float costanSh=cos(Shape[2][0])*(tan(Shape[3][0])*shapeSizev*2+shapeSizeh);
  return new float[][]{
    {shapeCenterX-costanSh+sinSv, shapeCenterY-sintanSh-cosSv},
    {shapeCenterX+costanSh+sinSv, shapeCenterY+sintanSh-cosSv},
    {shapeCenterX+cosSh-sinSv, shapeCenterY+sinSh+cosSv},
    {shapeCenterX-cosSh-sinSv, shapeCenterY-sinSh+cosSv}
  };
}

// 中心、サイズ、角度から台形の生成 2
float[][] makeTrapezoid2(float[][] Shape){
  float deno=1.5;
  float a=0;
  float b=0;
  float c=0;
  float d=0;
  float e=0;
  float f=0;
  float g=0;
  float h=0;
  float i=0;
  float j=0;
  float k=1;
  //Shape[2][1]=6;
  switch((int)Shape[2][1]) {
```

```
    case 1:
      c=1;d=1;//h=1;
      break;
    case 2:
      e=1;f=1;
      a=1;
      k=2;
      break;
    case 3:
      e=1;f=1;
      b=1;
      k=2;
      break;
    case 4:
      break;
    case 5:
      j=-1;
      break;
    case 6:
      j=1;
      break;
    default:
      e=1;f=1;
      break;
  }

  float per1=(deno+b)/(deno+a);
  float per2=(deno+a)/(deno+b);

  float shapeCenterX=Shape[0][0];
  float shapeCenterY=Shape[0][1];
  float shapeSizeh=Shape[1][0]/2;
  float shapeSizev=Shape[1][1]/2;
  float shapeAng=Shape[2][0];
  float sinSh=sin(shapeAng)*shapeSizeh;
  float cosSh=cos(shapeAng)*shapeSizeh;
  float sinSv=sin(shapeAng)*shapeSizev;
  float cosSv=cos(shapeAng)*shapeSizev;
  float sintanSh=sin(shapeAng)*tan(Shape[3][0]*k)*shapeSizev*2;
  float costanSh=cos(shapeAng)*tan(Shape[3][0]*k)*shapeSizev*2;
  if((Shape[2][1]==2)||(Shape[2][1]==3)){
    return new float[][]{
      {shapeCenterX-cosSh*(1-g)-costanSh*(1-c)*per1+sinSv*(per1-a/2),
       shapeCenterY-sinSh*(1-g)-sintanSh*(1-c)*per1-cosSv*(per1-a/2)},
      {shapeCenterX,
       shapeCenterY},
      {shapeCenterX+cosSh*(1-g)+costanSh*(1-d)*per2+sinSv*(per2-b/2),
       shapeCenterY+sinSh*(1-g)+sintanSh*(1-d)*per2-cosSv*(per2-b/2)},
      {shapeCenterX+cosSh+costanSh*(1-e)*(4-h)/4-sinSv,
       shapeCenterY+sinSh+sintanSh*(1-e)*(4-h)/4+cosSv},
      {shapeCenterX-cosSh-costanSh*(1-f)*(4-h)/4-sinSv,
       shapeCenterY-sinSh-sintanSh*(1-f)*(4-h)/4+cosSv}
    };
  }else if(Shape[2][1]<4){
    return new float[][]{
      {shapeCenterX-cosSh*(1-g)-costanSh*(1-c)*per1+sinSv*(per1-a/2),
       shapeCenterY-sinSh*(1-g)-sintanSh*(1-c)*per1-cosSv*(per1-a/2)},
      {shapeCenterX+cosSh*(1-g)+costanSh*(1-d)*per2+sinSv*(per2-b/2),
       shapeCenterY+sinSh*(1-g)+sintanSh*(1-d)*per2-cosSv*(per2-b/2)},
      {shapeCenterX+cosSh+costanSh*(1-e)*(4-h)/4-sinSv,
       shapeCenterY+sinSh+sintanSh*(1-e)*(4-h)/4+cosSv},
      {shapeCenterX-cosSh-costanSh*(1-f)*(4-h)/4-sinSv,
       shapeCenterY-sinSh-sintanSh*(1-f)*(4-h)/4+cosSv}
    };
  }else{
    return new float[][]{
      {shapeCenterX-cosSh+costanSh*j-sinSv*i,
       shapeCenterY-sinSh+sintanSh*j+cosSv*i},
      {shapeCenterX-cosSh*(1-g)-costanSh*(1-c)*per1+sinSv*(per1-a/2),
       shapeCenterY-sinSh*(1-g)-sintanSh*(1-c)*per1-cosSv*(per1-a/2)},
      {shapeCenterX+cosSh*(1-g)+costanSh*(1-d)*per2+sinSv*(per2-b/2),
       shapeCenterY+sinSh*(1-g)+sintanSh*(1-d)*per2-cosSv*(per2-b/2)},
      {shapeCenterX+cosSh+costanSh*j-sinSv*i,
       shapeCenterY+sinSh+sintanSh*j+cosSv*i},
      {shapeCenterX+cosSh+costanSh*(1-e)*(4-h)/4-sinSv,
       shapeCenterY+sinSh+sintanSh*(1-e)*(4-h)/4+cosSv},
      {shapeCenterX-cosSh-costanSh*(1-f)*(4-h)/4-sinSv,
       shapeCenterY-sinSh-sintanSh*(1-f)*(4-h)/4+cosSv}
    };
  }
}

// 中心、サイズ、角度から長方形の生成
float[][] makeRoadNode(float[][] Shape){
  float shapeCenterX=Shape[0][0];
  float shapeCenterY=Shape[0][1];
  float shapeSizeh=(Shape[1][0]+roadWidth)/2;
  float shapeSizev=(Shape[1][1]+roadWidth)/2;
  float sinSh=sin(Shape[2][0])*shapeSizeh;
  float cosSh=cos(Shape[2][0])*shapeSizeh;
  float sinSv=sin(Shape[2][0])*shapeSizev;
  float cosSv=cos(Shape[2][0])*shapeSizev;

  return new float[][]{
    {shapeCenterX+cosSh-sinSv, shapeCenterY+sinSh+cosSv},
    {shapeCenterX-sinSv, shapeCenterY+cosSv},
    {shapeCenterX-cosSh-sinSv, shapeCenterY-sinSh+cosSv}
  };
```

```
}

// 図形により分割され、なおかつ点を含む図形の生成
float[][] sliceShape(float[][] _Shape1, float[][] _Shape2){
  float[][] SlicedShape=_Shape1;
  for(int i=0; i<_Shape2.length; i++){
    float[][] _Segment;
    if(i>=_Shape2.length-1){
      _Segment=new float[][]{_Shape2[i], _Shape2[0]};
    }else{
      _Segment=new float[][]{_Shape2[i], _Shape2[i+1]};
    }
    SlicedShape=sliceShape(SlicedShape, _Segment, _Shape1[0]);
  }
  return SlicedShape;
}

// 直線により分割され、なおかつ点を含む図形の生成
float[][] sliceShape(float[][] _Shape, float[][] _Line, float[] _Point){
  float[][] SlicedShape1=new float[0][];
  float[][] SlicedShape2=new float[0][];
  float[][] CrrPoint=new float[0][];
  int[] CrrPTyps=new int[0];
  int crrPType=0;

  for(int i=0; i<_Shape.length; i++){
    float[][] _Segment=new float[0][];
    if(i>=_Shape.length-1){
      _Segment=new float[][]{_Shape[i], _Shape[0]};
    }else{
      _Segment=new float[][]{_Shape[i], _Shape[i+1]};
    }
    float[] _crrPoint=crrLtoSegPoint(_Line, _Segment);
    if(_crrPoint.length>0){
      CrrPoint=(float[][])append(CrrPoint, _Shape[i]);
      CrrPTyps=(int[])append(CrrPTyps, crrPType);
      CrrPoint=(float[][])append(CrrPoint, _crrPoint);
      CrrPTyps=(int[])append(CrrPTyps, 1);
      if(crrPType==0){
        crrPType=2;
      }else if(crrPType==2){
        crrPType=0;
      }
    }else{
      CrrPoint=(float[][])append(CrrPoint, _Shape[i]);
      CrrPTyps=(int[])append(CrrPTyps, crrPType);
    }
  }
  for(int i=0; i<CrrPoint.length; i++){
    if(CrrPTyps[i]==0){
      SlicedShape1=(float[][])append(SlicedShape1, CrrPoint[i]);
    }else if(CrrPTyps[i]==1){
      SlicedShape1=(float[][])append(SlicedShape1, CrrPoint[i]);
      SlicedShape2=(float[][])append(SlicedShape2, CrrPoint[i]);
    }else if(CrrPTyps[i]==2){
      SlicedShape2=(float[][])append(SlicedShape2, CrrPoint[i]);
    }
  }

  float[][] _SlicedShape=SlicedShape1;
  for(int i=0; i<SlicedShape1.length-1; i++){
    if((abs(SlicedShape1[i][0]-SlicedShape1[i+1][0])<0.0005)&&(abs(SlicedShape1[i][1]-SlicedShape1[i+1][1])<0.0005)){
      for(int j=0; j<_SlicedShape.length; j++){
        if((abs(SlicedShape1[i][0]-_SlicedShape[j][0])<0.0005)&&(abs(SlicedShape1[i][1]-_SlicedShape[j][1])<0.0005)){
          if(i==0){
            _SlicedShape=(float[][])subset(_SlicedShape, 1);
          }else if(i>=(_SlicedShape.length-1)){
            _SlicedShape=(float[][])shorten(_SlicedShape);
          }else{
            _SlicedShape=(float[][])concat((float[][])subset(_SlicedShape, 0, j), (float[][])subset(_SlicedShape, j+1));
          }
        }
        break;
      }
    }
  }
  SlicedShape1=_SlicedShape;
  _SlicedShape=SlicedShape2;
  for(int i=0; i<SlicedShape2.length-1; i++){
    if((abs(SlicedShape2[i][0]-SlicedShape2[i+1][0])<0.0005)&&(abs(SlicedShape2[i][1]-SlicedShape2[i+1][1])<0.0005)){
      for(int j=0; j<_SlicedShape.length; j++){
        if((abs(SlicedShape2[i][0]-_SlicedShape[j][0])<0.0005)&&(abs(SlicedShape2[i][1]-_SlicedShape[j][1])<0.0005)){
          if(i==0){
            _SlicedShape=(float[][])subset(_SlicedShape, 1);
          }else if(i>=(_SlicedShape.length-1)){
            _SlicedShape=(float[][])shorten(_SlicedShape);
          }else{
            _SlicedShape=(float[][])concat((float[][])subset(_SlicedShape, 0, j), (float[][])subset(_SlicedShape, j+1));
          }
        }
        break;
      }
    }
  }
  SlicedShape2=_SlicedShape;

  if(InShape(_Point, SlicedShape2)){
    return SlicedShape2;
```

```
  }else{
    return SlicedShape1;
  }
}

float[] crrLtoSegPoint(float[][] _Line, float[][] _Seg){
  float[] crrPoint=crrLinePoint(_Line, _Seg);

  if(crrPoint.length>0){
    if(crrLtoP(_Seg, crrPoint)){
      if((abs(crrPoint[0]-_Seg[1][0])<0.0005)&&((abs(crrPoint[1]-_Seg[1][1])<0.0005))){
        return new float[0];
      }else if((abs(crrPoint[0]-_Seg[0][0])<0.0005)&&((abs(crrPoint[1]-_Seg[0][1])<0.0005))){
        return new float[0];
      }else{
        return crrPoint;
      }
    }else{
      return new float[0];
    }
  }else{
    return new float[0];
  }
}

float[][] makeBBox(float[][] _Arr){
  float maxBBSizeX=0;
  float maxBBSizeY=0;
  float minBBSizeX=worldLenX;
  float minBBSizeY=worldLenY;
  for(int i=0; i<_Arr.length; i++){
    maxBBSizeX=max(maxBBSizeX, _Arr[i][0]);
    maxBBSizeY=max(maxBBSizeY, _Arr[i][1]);
    minBBSizeX=min(minBBSizeX, _Arr[i][0]);
    minBBSizeY=min(minBBSizeY, _Arr[i][1]);
  }

  return new float[][]{{minBBSizeX, minBBSizeY}, {minBBSizeX, maxBBSizeY},
                  {maxBBSizeX, maxBBSizeY}, {maxBBSizeX, minBBSizeY}};
}

float[][] voronoiSape(float[] Arr1, float[][] Arr2){
  float[][] basePlane=makeBBox(Arr2);
  for(int i=0; i<Arr2.length; i++){
    float[] halfPoint=centerPtoP(Arr1, Arr2[i]);
    float vAng=vectorAngle(Arr1, Arr2[i]);
    float[][] _Line=new float[][]{halfPoint, movePoint(halfPoint, 20, vAng+PI/2)};
    basePlane=sliceShape(basePlane, _Line, Arr1);
  }
  return basePlane;
}

float[][] vBaseShape(float[] Arr1, float[][] Arr2, float[][] Arr3){
  float[][] basePlane=Arr3;
  for(int i=0; i<Arr2.length; i++){
    float[][] _Line=orthSegment(Arr1, Arr2[i]);
    basePlane=sliceShape(basePlane, _Line, Arr1);
  }
  return basePlane;
}

float[][] orthSegment(float[] Arr1, float[] Arr2){
  float[] halfPoint=centerPtoP(Arr1, Arr2);
  float vAng=vectorAngle(Arr1, Arr2);
  return new float[][]{halfPoint, movePoint(halfPoint, 20, vAng)};
}

float[][] sinmplifyShape(float[][] _Arr){
  float[][] newArr=new float[][]{_Arr[0]};
  for(int i=1; i<_Arr.length; i++){
    if((_Arr[i][0]==newArr[newArr.length-1][0])&&(_Arr[i][1]==newArr[newArr.length-1][1])){
    }else{
      newArr=(float[][])append(newArr, _Arr[i]);
    }
  }
  if((newArr[0][0]==newArr[newArr.length-1][0])&&(newArr[0][1]==newArr[newArr.length-1][1])){
    newArr=(float[][])shorten(newArr);
  }

  return newArr;
}


class Camera{
  float et,rX,rZ;
  float det,drX,drZ;//default value;

  int projection=0;//perspective/ortho
  int control=1;//auto/manual
  float zN,zF;//Near,Far
  float top,bottom,right,left;//for ortho
  float fov,aspect;//for perspective

  float[] dtp=new float[3];
  float[] tp=new float[3];

  Camera(float[] _tp,float _et,float _rX,float _rZ){
    dtp=_tp;
```

```
  det=_et*2.1;
  drX=_rX;
  drZ=_rZ;
  adjustCam(dtp,det,drX,drZ);
}

  void resetCam(){
  tp=new float[]{worldLenX/2, worldLenY/2, 0};
  et=worldLen/2;
  rX=0;
  rX=0;
  rZ=0;
  drX=0.25*PI;
  drZ=0.25*PI;
  adjustCam(dtp,det,drX,drZ);
}

void adjustCam(float[] _tp,float _et,float _rX,float _rZ){
  tp=_tp;
  et=_et;
  rX=_rX;
  rZ=_rZ;
  updateCam();
}

void adjustCamPos(float[] _tp,float _et){
  tp=_tp;
  et=_et;
  updateCam();
}


void Display(){
  if(rX<0.) rX+=2*PI;
  else if(rX>=2*PI) rX-=2*PI;

  if(rZ<=0.){
    rX=90.;
    rZ=0.;
  }
  else if(rZ>=PI/2.0){
    rZ=PI/2.0;
  }

  float[] up=new float[]{0, 0, 0};
  if(rZ==0.) up[1]=1.;
  else up[2]=-1.;

  float[] eye=new float[]{0, 0, 0};
  eye[0]=tp[0]+et*cos(rX)*sin(rZ);
  eye[1]=tp[1]+et*sin(rX)*sin(rZ);
  eye[2]=tp[2]+et*cos(rZ);

  pg3D.camera(eye[0], eye[1], eye[2], tp[0], tp[1], tp[2], up[0], up[1], up[2]);

  if(projection==0) pg3D.perspective(fov,aspect,zN,zF);
  else if(projection==1) pg3D.ortho(left,right,bottom,top,zN,zF);
}

void projection(int _pm){
  projection=_pm;
  if(projection>2) projection=0;
  updateCam();
}

void shiftMode(){
  projection++;
  if(projection>2) projection=0;
  updateCam();
}

void control(int _cm){
  control=_cm;
  if(control>2)control=0;
  updateCam();
}

void shiftControl(){
  control++;
  if(control>2)control=0;
  updateCam();
}

int getControl(){
  return control;
}

void updateCam(){
  fov= 50*PI/180.;
  aspect=float(width-100)/float(height);

  zN=0.0001;
  zF=worldLen*4;

  top=et*tan(fov/2.);
  bottom=-top;
  right=top*aspect;
  left=-right;
```

```
  }

  void zoom(float _dt){
    et+=_dt;
    updateCam();
  }

  void rotation(float _drX,float _drZ){
    rX+=_drX;
    rZ+=_drZ;
    updateCam();
  }

  void drag(float _tpX,float _tpY){
    if(_tpX<0){
      tp[0]-=_tpY*cos(rX)*cos(rZ)+abs(_tpX)*cos(rX+PI/2);
      tp[1]-=_tpY*sin(rX)*cos(rZ)+abs(_tpX)*sin(rX+PI/2);
      tp[2]+=_tpY*sin(rZ);
    }else{
      tp[0]-=_tpY*cos(rX)*cos(rZ)+abs(_tpX)*cos(rX-PI/2);
      tp[1]-=_tpY*sin(rX)*cos(rZ)+abs(_tpX)*sin(rX-PI/2);
      tp[2]+=_tpY*sin(rZ);
    }
  }
}

import controlP5.*;

ControlP5 controlP5;

CheckBox checkbox;

public float sHouseWidth, sHouseDepth, sGardenDepth, sGardenWidth, sGardenAngle, sHouseSpan;

void ControlP5Setup(){
  controlP5 = new ControlP5(this);
  controlP5.setColorForeground(color(80, 80, 80));
  controlP5.setColorBackground(color(150, 150, 150));
  sHouseWidth=rctSizeX;
  sHouseDepth=rctSizeY;
  sGardenDepth=gdnSizeY;
  sGardenWidth=gdnSizeX;
  sGardenAngle=gdnAng*180/PI;
  sHouseSpan=shapeBtw-rctSize;

  controlP5.addButton("Main",0, 20, 20, 40, 20);

  controlP5.addButton("addHouse", 0, width-360, height-130, 70, 20);
  controlP5.addButton("removeHouse", 0, width-280, height-130, 70, 20);

  controlP5.addSlider("HouseWidth", 0, 20, sHouseWidth, width-185, 15, 120, 15).setId(1);
  controlP5.addSlider("HouseDepth", 0, 20, sHouseDepth, width-185, 45, 120, 15).setId(2);
  controlP5.addSlider("GardenWidth", 0, 20, sGardenWidth, width-185, 75, 120, 15).setId(3);
  controlP5.addSlider("GardenDepth", 0, 50, sGardenDepth, width-185, 105, 120, 15).setId(4);
  controlP5.addSlider("GardenAngle", 0, 45, sGardenAngle, width-185, 135, 120, 15).setId(5);
  controlP5.addSlider("HouseSpan", 0, 10, sHouseSpan, width-185, 165, 120, 15).setId(6);

  checkbox = controlP5.addCheckBox("checkBox", width-86, height-250);
  checkbox.addItem("FrameAdvance", 0);
  checkbox.setColorForeground(color(180));
  checkbox.setColorActive(color(10));

  controlP5.addButton("Reflect",2, width-185, height-230, 60, 20);
  controlP5.addButton("Stop",1, width-110, height-230, 40, 20);
  controlP5.addButton("Start",2, width-50, height-230, 40, 20);
}

public void Main(int theValue){
  if(dispMode==2){
    dispMode=3;
  }else if(dispMode==3){
    dispMode=2;
  }
  addManual=false;
  pressButton=true;
}

public void addHouse(int theValue){
  if((dispMode==3)&&(!goRunFlg)){
    addManual=true;
    pressButton=true;
  }
}

public void removeHouse(int theValue){
  if((!goRunFlg)&&(addManual)){
    addedHouse=false;
    pressButton=true;
  }
}

public void Stop(int theValue){
  goRunFlg=false;
  addManual=false;
  addedHouse=false;
}

public void Start(int theValue){
```

```
  goRunFlg=true;
  addManual=false;
}

public void HouseWidth(int theValue){
  sHouseWidth=theValue;
}

public void HouseDepth(int theValue){
  sHouseDepth=theValue;
}

public void GardenDepth(int theValue){
  sGardenDepth=theValue;
}

public void GardenWidth(int theValue){
  sGardenWidth=theValue;
}

public void GardenAngle(int theValue){
  sGardenAngle=theValue;
}

public void HouseSpan(int theValue){
  sHouseSpan=theValue;
}

public void Reflect(int theValue){
  if(!goRunFlg){
    rctSizeX=sHouseWidth;
    rctSizeY=sHouseDepth;
    gdnSizeY=sGardenDepth;
    gdnSizeX=sGardenWidth;
    gdnAng=sGardenAngle*PI/180;
    shapeBtw=sqrt(sq(rctSizeX)+sq(rctSizeY))+sHouseSpan;
  }
}

void controlEvent(ControlEvent theEvent){
  if(theEvent.isGroup()){
    if((int)theEvent.group().arrayValue()[0]==1) {
      frameAdvFlg=true;
      goRunFlg=false;
    }else{
      frameAdvFlg=false;
      goRunFlg=true;
    }
  }
}

void keyPressed() {
  switch(key){
    case '1':
      if(houseSelectFlg){
        if(houseInfoMode==1){
          houseInfoMode=-1;
        }else{
          houseInfoMode=1;
        }
      }
      break;
    case '2':
      if(houseSelectFlg){
        if(houseInfoMode==2){
          houseInfoMode=-1;
        }else{
          houseInfoMode=2;
        }
      }
      break;
    case '3':
      if(houseSelectFlg){
        if(houseInfoMode==3){
          houseInfoMode=-1;
        }else{
          houseInfoMode=3;
        }
      }
      break;
    case '4':
      if(houseSelectFlg){
        if(houseInfoMode==4){
          houseInfoMode=-1;
        }else{
          houseInfoMode=4;
        }
      }
      break;
    case '5':
      if(houseSelectFlg){
        if(houseInfoMode==5){
          houseInfoMode=-1;
        }else{
          houseInfoMode=5;
        }
      }
      break;
```

```
    case 'v':
      if(houseSelectFlg){
        if(houseInfoMode==6){
          houseInfoMode=-1;
        }else{
          houseInfoMode=6;
        }
      }
      break;
    case 'c':
      if(houseSelectFlg){
        if(houseInfoMode==7){
          houseInfoMode=-1;
        }else{
          houseInfoMode=7;
        }
      }
      break;
    case 'n':
      if(houseSelectFlg){
        if(houseInfoMode==8){
          houseInfoMode=-1;
        }else{
          houseInfoMode=8;
        }
      }
      break;
    case 'a':
      break;
    case 'p':
      break;
    case 'e':
      exportFlg=true;
      break;
    case 'r':
      break;
    case 'Q':
      areaRate.flush();
      areaRate.close();
      gdnType.flush();
      gdnType.close();
      sunshine.flush();
      sunshine.close();
      vArea.flush();
      vArea.close();
      for(int i=0; i<appVoid.length; i++){
        appVoid[i].flush();
        appVoid[i].close();
      }
      exit();
      break;
    case 'D':
      dispMode++;
      if(dispMode>1)dispMode=0;
      cam.updateCam();
      Display3D();
      break;
    case 'X':
      cam.adjustCamPos(cam.tp, worldLen);
      cam.updateCam();
      Display3D();
    case 'A':
      cam.resetCam();
      cam.updateCam();
      Display3D();
      break;
    case 'C':
      cam.adjustCamPos(new float[]{worldLenX/2, 0, worldLenY/2}, cam.et);
      cam.updateCam();
      Display3D();
      break;
  }
}

void mouseDragged(){
  boolean conFlg1=false;
  boolean conFlg2=false;
  if(dispMode==0){
    conFlg2=true;
  }else if(dispMode==1){
    conFlg1=true;
  }else if(dispMode==2){
    if((mouseX<width-200)&&(mouseY<height-100)&&(pmouseX<width-200)&&(pmouseY<height-100)){
      conFlg1=true;
    }
    if((mouseX>width-200)&&(mouseY>height-200)&&(pmouseX>width-200)&&(pmouseY>height-200)){
      conFlg2=true;
    }
  }else if(dispMode==3){
    if((mouseX<width-200)&&(mouseY<height-100)&&(pmouseX<width-200)&&(pmouseY<height-100)){
      conFlg2=true;
    }
    if((mouseX>width-200)&&(mouseY>height-200)&&(pmouseX>width-200)&&(pmouseY>height-200)){
      conFlg1=true;
    }
  }
  if((dispMode!=0)&&(mouseButton==RIGHT)&&(conFlg1)){
    if(keyPressed&&keyCode==SHIFT){
```

```
      cam.drag((mouseX-pmouseX)*1., (mouseY-pmouseY)*1.);
      cam.updateCam();
      Display3D();
    }else if(keyPressed&&keyCode==CONTROL){
      cam.zoom((mouseY-pmouseY)*25.);
      cam.updateCam();
      Display3D();
    }else{
      cam.rotation((mouseX-pmouseX)/100.0, -(mouseY-pmouseY)/100.0);
      cam.updateCam();
      Display3D();
    }
  }

  if((dispMode==3)&&(mouseButton==LEFT)&&(conFlg2)&&(addManual)){
    houseDragged=true;
  }
}

void mousePressed(){
  if((dispMode==3)&&(!goRunFlg)&&(addManual)){
    if((mouseX<width-200)&&(mouseY<height-100)){
      if(!pressButton){
        pressMouseX=mouseX-marginX;
        pressMouseY=mouseY-marginY;
      }
    }
  }
}

void mouseReleased(){
  if((dispMode==3)&&(!goRunFlg)&&(addManual)){
    if(pressButton){
      houseDragged=false;
      pressButton=false;
    }else{
      if((pressMouseX<width-200)&&(pressMouseY<height-100)){
        addedHouse=true;
        houseDragged=false;
      }
    }
  }
}

void mouseMoved(){
  if(addManual){
    fill(0, 100, 255, 220);
    stroke(0, 0, 255);
    rect(mouseX-rctSizeX/2, mouseY-rctSizeY/2, rctSizeX, rctSizeY);
  }
}


// 図形の描画
void drawShape(float[][] Arr){
  beginShape();
  for(int i=0; i<Arr.length; i++){
    vertex(Arr[i][0]+marginX, Arr[i][1]+marginY);
  }
  endShape(CLOSE);
}

void drawShape(float[][] Arr, PGraphics _pg){
  _pg.beginShape();
  for(int i=0; i<Arr.length; i++){
    _pg.vertex(Arr[i][0]+marginX, Arr[i][1]+marginY);
  }
  _pg.endShape(CLOSE);
}

// 画面の描画
void Display(){
  // 画面の初期化
  background(220);
  noStroke();
  fill(200, 255, 200);
  rect(-worldLenX*3,-worldLenY*3,worldLenX*6,worldLenY*6);

  for(int i=0; i<5; i++){
    switch(i){
      case 0:
        fill(100, 200, 100);
        noStroke();
        if(Area.length>0)drawShape(Area);
        break;
      case 1:
        if(Lines.size()>0){
          for(int j=0; j<Lines.size(); j++){
            float[][] t_road=(float[][])Lines.get(j);
            stroke(66);
            strokeWeight(roadWidth);
            boolean flg5=false;
            for(int k=0; k<Lines.size(); k++){
              float[][] tt_road=(float[][])Lines.get(k);
              if(tt_road[2][1]==j){
                flg5=true;
                break;
              }
```

```
          }
          drawShape(new float[][]{t_road[0], t_road[1]});
        }
      }
      break;
    case 2:
      if(Houses.size()>0){
        for(int j=0; j<Houses.size(); j++){
          House t_House=(House)Houses.get(j);
          int t_life=(int)t_House.life;
          fill(150-t_life*(150/lifeWidth[1]), 100, t_life*(255/lifeWidth[1]), 180);
          stroke(150-t_life*(100/lifeWidth[1]), 100, 100+t_life*(155/lifeWidth[1]));
          strokeWeight(0.5);
          drawShape(t_House.makeShape());
          /*
          fill(255);
          textFont(font9);
          text(t_House.id, t_House.posX, t_House.posY);
          */
          /*
          fill(255, 0, 0);
          stroke(255, 0, 0);
          ellipse(makeRoadNode(t_house)[0][0]+marginX, makeRoadNode(t_house)[0][1]+marginY,2,2);
          ellipse(makeRoadNode(t_house)[1][0]+marginX, makeRoadNode(t_house)[1][1]+marginY,2,2);
          ellipse(makeRoadNode(t_house)[2][0]+marginX, makeRoadNode(t_house)[2][1]+marginY,2,2);
          */
        }
      }
      break;
    case 3:
      fill(0, 60, 60, 100);
      stroke(0, 255, 0, 100);
      strokeWeight(0.5);
      if(UniSpaces.size()>0){
        for(int j=0; j<UniSpaces.size(); j++){
          float[][] t_garden=(float[][])UniSpaces.get(j);
          drawShape(makeTrapezoid2(t_garden));
        }
      }
      break;
    case 4:
      if(Houses.size()>0){
        for(int j=0; j<Houses.size(); j++){
          House t_House=(House)Houses.get(j);
          //fill(255, 0, 0, 50);
          noFill();
          stroke(255, 150, 0, 180);
          strokeWeight(1);
          drawShape(t_House.getVoronoi());
        }
      }
      break;
    case 5:
      if(Houses.size()>0){
        for(int j=0; j<Houses.size(); j++){
          if(j==Houses.size()-1){
            House t_House=(House)Houses.get(j);
            //fill(255, 0, 0, 50);
            noFill();
            stroke(0, 255, 0);
            strokeWeight(1);
            if(t_House.Neighbors.length>0){
              for(int k=0; k<t_House.Neighbors.length; k++){
                //println(t_House.id+" "+k+" "+t_House.Neighbors.length+" "+t_House.Neighbors[k]+" "+searchId(t_House.Neighbors[k]));
                int targetID=searchId(t_House.Neighbors[k]);
                if(targetID!=-1){
                  House tt_House=(House)Houses.get(searchId(t_House.Neighbors[k]));
                  drawShape(new float[][]{t_House.getPoint(), tt_House.getPoint()});
                }else{
                  println(t_House.id+" "+t_House.Neighbors[k]+" noooooo!");
                  goRunFlg=false;
                }
              }
            }
          }
        }
      }
      break;
  }
  fill(0);
  noStroke();
  strokeWeight(0.5);
  //rect(10+20*i, 10, 10, 20);
 }
}

void Display(PGraphics _pg, int _drawNum, boolean forPrint, boolean forPDF){
 // 画面の初期化
 _pg.beginDraw();
 _pg.background(220);
 _pg.noStroke();
 _pg.fill(200, 255, 200);
 _pg.rect(-worldLenX*3,-worldLenY*3,worldLenX*6,worldLenY*6);

 for(int i=0; i<_drawNum; i++){
  switch(i){
   case 0:
    _pg.fill(100, 200, 100);
```

```
      _pg.noStroke();
      if(Area.length>0)drawShape(Area, _pg);
      break;
    case 1:
      if(Lines.size()>0){
        for(int j=0; j<Lines.size(); j++){
          float[][] t_road=(float[][])Lines.get(j);
          _pg.stroke(66);
          _pg.strokeWeight(roadWidth);
          boolean flg5=false;
          for(int k=0; k<Lines.size(); k++){
            float[][] tt_road=(float[][])Lines.get(k);
            if(tt_road[2][1]==j){
              flg5=true;
              break;
            }
          }
          drawShape(new float[][]{t_road[0], t_road[1]}, _pg);
        }
      }
      break;
    case 2:
      if(Houses.size()>0){
        for(int j=0; j<Houses.size(); j++){
          House t_House=(House)Houses.get(j);
          int t_life=(int)t_House.life;
          _pg.fill(150-t_life*(105/lifeWidth[1]), 100, t_life*(255/lifeWidth[1]), 220);
          _pg.stroke(150-t_life/3, 0, t_life/1.5);
          _pg.strokeWeight(0.5);
          drawShape(t_House.makeShape(), _pg);
        }
      }
      break;
    case 3:
      _pg.fill(0, 60, 60, 100);
      _pg.stroke(0, 255, 0, 100);
      _pg.strokeWeight(0.5);
      if(UniSpaces.size()>0){
        for(int j=0; j<UniSpaces.size(); j++){
          float[][] t_garden=(float[][])UniSpaces.get(j);
          drawShape(makeTrapezoid2(t_garden), _pg);
        }
      }
      break;
    case 4:
      if(Houses.size()>0){
        for(int j=0; j<Houses.size(); j++){
          House t_House=(House)Houses.get(j);
          int t_life=(int)t_House.life;
          //_pg.fill(255, 0, 0, 50);
          _pg.noFill();
          _pg.stroke(255, 150, 0, 180);
          _pg.strokeWeight(1);
          drawShape(t_House.getVoronoi(), _pg);
        }
      }
      break;
    case 5:
      if(Houses.size()>0){
        for(int j=0; j<Houses.size(); j++){
          if(j==Houses.size()-1){
            House t_House=(House)Houses.get(j);
            //_pg.fill(255, 0, 0, 50);
            _pg.noFill();
            _pg.stroke(0, 255, 0);
            _pg.strokeWeight(1);
            if(t_House.Neighbors.length>0){
              for(int k=0; k<t_House.Neighbors.length; k++){
                //println(t_House.id+" "+k+" "+t_House.Neighbors.length+" "+t_House.Neighbors[k]+" "+searchId(t_House.Neighbors[k]));
                int targetID=searchId(t_House.Neighbors[k]);
                if(targetID!=-1){
                  House tt_House=(House)Houses.get(searchId(t_House.Neighbors[k]));
                  drawShape(new float[][]{t_House.getPoint(), tt_House.getPoint()}, _pg);
                }else{
                  println(t_House.id+" "+t_House.Neighbors[k]+" noooooooo!");
                  goRunFlg=false;
                }
              }
            }
          }
        }
      }
      break;
  }
  if(forPrint){
    _pg.fill(0);
    _pg.noStroke();
    _pg.strokeWeight(0.5);
    _pg.rect(10+20*i, 10, 10, 20);
  }
}
if(forPrint){
  _pg.fill(0);
  _pg.noStroke();
  _pg.strokeWeight(0.5);
  _pg.rect(0, height-100, width, 100);
  _pg.fill(255);
  _pg.noStroke();
```

```
    _pg.textFont(font12);
    _pg.text(message1 ,width-250, height-68);
    _pg.text(message2 ,width-250, height-36);
    _pg.text(message3 ,20, height-68);
    _pg.text(message4 ,20, height-36);
    if(forPDF){
      _pg.dispose();
    }
  }
  _pg.endDraw();
}

void textDisplay(){
  pgText.beginDraw();
  pgText.fill(0);
  pgText.noStroke();
  pgText.strokeWeight(0.5);
  pgText.rect(0, 0, width, 100);
  pgText.fill(255);
  pgText.noStroke();
  pgText.textFont(font12);
  pgText.text(message1 ,width-250, 32);
  pgText.text(message2 ,width-250, 64);
  pgText.text(message3 ,20, 32);
  pgText.text(message4 ,20, 64);
  pgText.endDraw();
}

void drawShape3D(float[][] _Arr){
  pg3D.beginShape();
  for(int i=0; i<_Arr.length; i++){
    pg3D.vertex(_Arr[i][0]+marginX, _Arr[i][1]+marginY, 0);
  }
  pg3D.endShape(CLOSE);
}

void drawShape3D(float[][] _Arr, float _z){
  pg3D.beginShape();
  for(int i=0; i<_Arr.length; i++){
    pg3D.vertex(_Arr[i][0]+marginX, _Arr[i][1]+marginY, _z);
  }
  pg3D.endShape(CLOSE);
}

void drawExShape3D(float[][] _Arr){
  pg3D.beginShape();
  for(int i=0; i<_Arr.length; i++){
    pg3D.vertex(_Arr[i][0]+marginX, _Arr[i][1]+marginY, 0);
  }
  pg3D.endShape(CLOSE);
  pg3D.beginShape();
  for(int i=0; i<_Arr.length; i++){
    pg3D.vertex(_Arr[i][0]+marginX, _Arr[i][1]+marginY, 9);
  }
  pg3D.endShape(CLOSE);
  for(int i=0; i<_Arr.length; i++){
    int _i=i+1;
    if(_i>=_Arr.length)_i=0;
    pg3D.beginShape();
    pg3D.vertex(_Arr[i][0]+marginX, _Arr[i][1]+marginY, 0);
    pg3D.vertex(_Arr[_i][0]+marginX, _Arr[_i][1]+marginY, 0);
    pg3D.vertex(_Arr[_i][0]+marginX, _Arr[_i][1]+marginY, 9);
    pg3D.vertex(_Arr[i][0]+marginX, _Arr[i][1]+marginY, 9);
    pg3D.endShape(CLOSE);
  }
}

void Display3D(){
  // 画面の初期化
  pg3D.beginDraw();
  pg3D.background(220);
  pg3D.noStroke();
  pg3D.fill(200, 255, 200);
  pg3D.beginShape();
  pg3D.vertex(-worldLenX*3,-worldLenY*3, -1);
  pg3D.vertex(worldLenX*3,-worldLenY*3, -1);
  pg3D.vertex(worldLenX*3,worldLenY*3, -1);
  pg3D.vertex(-worldLenX*3,worldLenY*3, -1);
  pg3D.endShape(CLOSE);
  axis.Display();

  for(int i=0; i<5; i++){
    switch(i){
      case 0:
        //pg3D.noFill();
        pg3D.fill(100, 200, 100);
        pg3D.noStroke();
        drawShape3D(Area);
        break;
      case 1:
        for(int j=0; j<Lines.size(); j++){
          float[][] t_road=(float[][])Lines.get(j);
          pg3D.stroke(66);
          pg3D.strokeWeight(roadWidth);
          boolean flg5=false;
          for(int k=0; k<Lines.size(); k++){
            float[][] tt_road=(float[][])Lines.get(k);
            if(tt_road[2][1]==j){
```

```
            flg5=true;
            break;
          }
        }
        //if((distPtoP(t_road[0], t_road[1])>rctSize/sqrt(2))||(flg5))
          drawShape3D(new float[][]{t_road[0], t_road[1]});
      }
      break;
    case 2:
      for(int j=0; j<Houses.size(); j++){
        House t_House=(House)Houses.get(j);
        int t_life=(int)t_House.life;
        pg3D.fill(150-t_life*(105/lifeWidth[1]), 100, t_life*(255/lifeWidth[1]), 220);
        pg3D.stroke(150-t_life/3, 0, t_life/1.5);
        pg3D.strokeWeight(0.5);
        drawExShape3D(t_House.makeShape());
      }
      break;
    case 3:
      pg3D.fill(0, 60, 60, 100);
      pg3D.stroke(0, 255, 0, 100);
      pg3D.strokeWeight(0.5);
      for(int j=0; j<UniSpaces.size(); j++){
        float[][] t_garden=(float[][])UniSpaces.get(j);
        drawShape3D(makeTrapezoid2(t_garden), 0.1);
      }
      break;
    case 4:
      if(Houses.size()>0){
        for(int j=0; j<Houses.size(); j++){
          House t_House=(House)Houses.get(j);
          int t_life=(int)t_House.life;
          pg3D.noFill();
          pg3D.stroke(255, 150, 0, 180);
          pg3D.strokeWeight(1);
          drawShape3D(t_House.getVoronoi());
        }
      }
      break;
  }
 }
 cam.updateCam();
 cam.Display();
 pg3D.endDraw();
}

void initSetting(){
 pg2D=createGraphics(width-200, height-100, JAVA2D);
 pgText=createGraphics(width-200, 200, JAVA2D);
 pg3D=createGraphics(width-200, height-100, P3D);
 pgFP=createGraphics(width-200, height, P3D);
 pdf=createGraphics(width-200, height, P3D);

 scaleDis=1;
 worldLenX=width-200;
 worldLenY=height-100;
 worldLen=(worldLenX+worldLenY)/2;

 //ArrayList の初期化
 Lines=new ArrayList();
 UniSpaces=new ArrayList();
 Neighbors=new ArrayList();
 Houses=new ArrayList();
 houseID=0;
 HouseIDs=new int[0];
 revHouseIDs=new int[0];

 // ファイル書き出し用のフラグ
 exportFlg=false;
 fileNum=1;

 goRunFlg=false;
 frameAdvFlg=false;
 addManual=false;
 addedHouse=false;
 houseDragged=false;
 pressButton=false;
 houseSelectFlg=false;

 houseInfoMode=-1;

 dragAng=PI;

 loadGeo();

 // 敷地の設定 ( 横と縦の最大値 )
 float maxAreaSizeX=0;
 float maxAreaSizeY=0;
 float minAreaSizeX=worldLenX;
 float minAreaSizeY=worldLenY;
 for(int i=0; i<Area.length; i++){
  maxAreaSizeX=max(maxAreaSizeX, Area[i][0]);
  maxAreaSizeY=max(maxAreaSizeY, Area[i][1]);
  minAreaSizeX=min(minAreaSizeX, Area[i][0]);
  minAreaSizeY=min(minAreaSizeY, Area[i][1]);
 }
 areaSizeX=maxAreaSizeX+minAreaSizeX;
 areaSizeY=maxAreaSizeY+minAreaSizeY;
```

```
  // 敷地をウィンドウに中央表示するための変数を設定
  marginX=(worldLenX-areaSizeX*scaleDis)/2;
  marginY=(worldLenY-areaSizeY*scaleDis)/2;

  // 家の縦横の大きさを指定
  rctSizeX=12;
  rctSizeY=7.5;
  rctSize=sqrt(sq(rctSizeX)+sq(rctSizeY));
  shapeBtw=rctSize+1.5;
  // 庭の縦横の大きさを指定
  gdnAng=PI*12/180;
  gdnSizeX=2;
  gdnSizeY=22.5;
  gdnSize=sqrt(sq(gdnSizeX+tan(gdnAng)*gdnSizeY)+sq(gdnSizeY));
  gdnShapeVar=7;
  roadWidth=4;

  grid=new int[20][20];
  for(int i=0; i<grid.length; i++){
    for(int j=0; j<grid[0].length; j++){
      grid[i][j]=0;
    }
  }

  t=0;
  HperY=5;
  lifeWidth=new int[]{40, 60};

  font16=createFont("MS-Gothic", 16);
  font12=createFont("MS-Gothic", 12);
  font9=createFont("MS-Gothic", 9);
  font6=createFont("MS-Gothic", 6);
  textMode(MODEL);
  setMessage();

  areaRate=createWriter("export/areaRate.tsv");
  gdnType=createWriter("export/gdnType.tsv");
  sunshine=createWriter("export/sunshine.tsv");
  vArea=createWriter("export/vArea.tsv");
  witerName=new String[]{"appVoid1", "appVoid2", "appVoid3", "appVoid4", "appVoid5"};
  appVoid=new PrintWriter[witerName.length];
  for(int i=0; i<appVoid.length; i++){
    appVoid[i]=createWriter("export/"+witerName[i]+".tsv");
  }

  dispMode=3;

  axis=new Axis(worldLen);
  cam=new Camera(new float[]{worldLenX/2, worldLenY/2, 0}, worldLen/2, PI*6/24, PI*8/24);
  count=0;

  Display3D();
  Display();
  textDisplay();
}


void loadGeo(){
  String[] SiteTable=loadStrings("import/site.tsv");
  if(SiteTable.length==0) return;
  String[][] data=new String[0][];
  for(int i=0; i<SiteTable.length; i++){
    String[] PointTable = split(SiteTable[i], TAB);
    data=(String[][])append(data, PointTable);
  }
  // 敷地の設定
  Area=new float[0][];
  for(int i=0; i<data.length; i++){
    Area=(float[][])append(Area, new float[]{parseFloat(data[i][0]), parseFloat(data[i][1])});
  }
  //Area=new float[][]{{50, 250}, {250, 250}, {250, 50}, {50, 50}};

  String[] RoadTable=loadStrings("import/road.tsv");
  if(RoadTable.length==0) return;
  data=new String[0][];
  for(int i=0; i<RoadTable.length; i++){
    String[] PointTable=split(RoadTable[i], TAB);
    data=(String[][])append(data, PointTable);
  }
  for(int i=0; i<data.length; i++){
    float[][] road={{parseFloat(data[i][0]), parseFloat(data[i][1])}, {parseFloat(data[i][2]), parseFloat(data[i][3])}, {1, -2}};
    Lines.add(road);
  }

  // 敷地の外周を道路として登録
  for(int i=0; i<Area.length; i++){
    float[][] t_road={Area[i], Area[(i+1)%Area.length]};
    Lines.add(new float[][]{t_road[0], t_road[1], {1, -2}});
  }
}

public class culcThread implements Runnable{
  public void run(){
    while(true){
      if(goRunFlg){
        if(count>=moviefps){
          count=0;
```

```
float[][] road=new float[3][2];
float[][] road2=new float[3][2];
float[][] road3=new float[3][2];
float[][] garden=new float[3][2];
float[][] neighbor=new float[2][2];
float[][] pgarden=new float[3][2];
int[] CHouseId=new int[0];
float[] DistHouse=new float[0];
House House=new House();

int roadFlg=0;
boolean addFlg=true;
boolean[] FLG=new boolean[]{true, true, false};

while(true){
 // フラグ等の初期化
 int m=0;
 //boolean[] FLG=new boolean[]{true, true, false};
 boolean flg1=FLG[0];
 boolean flg2=FLG[1];
 boolean flg3=FLG[2];
 int t_m=m;
 boolean textFlg=false;
 // 新しく家の置ける場所を探す
 while(true){
  //delay(500);
  if(m-t_m>50){
   //print(m+", ");
   t_m=m;
  }
  //while を回し過ぎるとプログラムを停止する
  if((m>100)&&(m>2*Houses.size())){
   addFlg=false;
   break;
  }
  if(!goRunFlg){
   addFlg=false;
   break;
  }
  if(addedHouse){
   addedHouse=false;
   textFlg=true;
   House=new House(pressMouseX, pressMouseY, houseID);
   House.dirA=dragAng;
  }else{
   // 家が 0 のときはランダムな位置に置く
   if(Houses.size()<1){
    House=new House(random(areaSizeX-rctSizeX), random(areaSizeY-rctSizeY), houseID);
   // 家があるときは既存の家をランダムに抽出してその近くで場所を探す
   }else if(Houses.size()<260){
    int logm=6-(int)(log(m)/log(10));
    House t_House=(House)Houses.get((int)random(Houses.size()));
    House=new House(t_House.posX+rctSizeX*random(logm*2)-rctSizeX*logm, t_House.posY+rctSizeY*random(logm*2)-rctSizeY*logm, houseID);
   }else if(Houses.size()<270){
    int logm=4-(int)(m/1000);
    House t_House=(House)Houses.get((int)random(Houses.size()));
    House=new House(t_House.posX+rctSizeX*random(logm*2)-rctSizeX*logm, t_House.posY+rctSizeY*random(logm*2)-rctSizeY*logm, houseID);
   }else{
    int logm=3-(int)(m/1000);
    House t_House=(House)Houses.get((int)random(Houses.size()));
    House=new House(t_House.posX+rctSizeX*random(logm*2)-rctSizeX*logm, t_House.posY+rctSizeY*random(logm*2)-rctSizeY*logm, houseID);
   }
  }
  garden=new float[][]{movePoint(House.getPoint(), (rctSizeY+gdnSizeY)/2, House.dirA),{gdnSizeX, gdnSizeY}, {House.dirA, 0}, {gdnAng}};

  // 家の角度を変更するための for 文
  for(int k=1; k<=13; k++){
   // 左右交互に家の角度を振る
   int pam=(k%2)*2-1;
   House.dirA=House.dirA+PI*(k-1)*6*pam/180;
   // 家の情報を一時変数に代入
    House n_House=House;
     float[][] n_garden=new float[][]{movePoint(n_House.getPoint(), (rctSizeY+gdnSizeY)/2, n_House.dirA),{gdnSizeX, gdnSizeY}, {n_House.dirA, 0},
{gdnAng}};
   // 家の場所を微妙に調整するための for 文
   for(int j=0; j<rctSize*2; j++){
    // 庭の情報を生成して一時変数に代入
    // フラグの初期化
    flg1=FLG[0];
    flg2=FLG[1];
    flg3=FLG[2];
    // 家が敷地の中に納まっているかの判定
    if(InShape(n_House.makeShape(), Area)&&InShape(makeTrapezoid2(n_garden), Area)){
     // 既存の家が 0 のときは ArrayOutofBounds エラーがでないように除外
     if(Houses.size()<1){
      flg3=true;
      addFlg=true;
      break;
     }
     // 庭の形を複数通り試すための for 文
     for(int l=0; l<gdnShapeVar; l++){
      // フラグの初期化
      flg1=FLG[0];
      flg2=FLG[1];
      flg3=FLG[2];
      n_garden[2][1]=l;
      CHouseId=new int[0];
```

```
        DistHouse=new float[0];
        // 既存の各家との関係を調べるための for 文
        for(int i=0; i<Houses.size(); i++){
          House t_House=(House)Houses.get(i);
          float[][] t_garden=(float[][])UniSpaces.get(i);
          float distHtoH=n_House.distHouse(t_House.posX, t_House.posY);
          CHouseId=(int[])append(CHouseId, t_House.id);
          DistHouse=(float[])append(DistHouse, distHtoH);
          // 近くにある家との関係だけを見るための条件文
          if((n_House.distHouse(t_garden[0])<(rctSize+gdnSize/2))||(t_House.distHouse(n_garden[0])<(rctSize+gdnSize/2))){
            if((flg1)&&(flg2)){
              // 既存の家と重なっていないかの判定
              if(shapeBtw>=rctSize){
                if(distHtoH<shapeBtw)flg1=false;
              }else{
                if(crrShape(n_House.makeShape(), t_House.makeShape())){
                  flg1=false;
                }
              }
              if((flg1)&&(flg2)){
                // 既存の家の庭と重なっていないかの判定
                if(crrShape(n_House.makeShape(), makeTrapezoid2(t_garden))){
                  flg1=false;
                }
              }
            }
            if((flg1)&&(flg2)){
              // 庭が既存の家と重なっていないかの判定
              if(crrShape(makeTrapezoid2(n_garden), t_House.makeShape())){
                flg2=false;
              }
            }
            if((flg1)&&(flg2)&&(!flg3)){
              // 庭が既存の庭と重なっているかの判定
              if(crrShape(makeTrapezoid2(n_garden), makeTrapezoid2(t_garden)))flg3=true;
              //flg3=true;
            }
          }
        }
        // 家が道路と重なっていないかの判定
        if((flg1)&&(flg2)&&(flg3)){
          for(int i=0; i<Lines.size(); i++){
            float[][] t_road=(float[][])Lines.get(i);
            // 近くの道との関係だけを判定
            if(distLtoP(t_road, n_House.getPoint())<rctSize*2){
              // 家が道路と重なっていないかの判定
              if(crrShape(n_House.makeShape(), t_road[0],t_road[1]))flg1=false;
              // 庭が道路と重なっていないかの判定
              //if(crrShape(makeTrapezoid2(n_garden), t_road[0],t_road[1]))flg1=false;
            }
          }
        }
        // 庭の向きが決定したらループを抜ける
        if((flg1)&&(flg2)&&(flg3))break;
      }
      // 敷地に収まっていないときは上の処理を飛ばす
      }else{
        //if(textFlg)println("Why?");
        flg1=false;
        flg2=false;
        flg3=false;
        continue;
      }
      // 家の位置が決まったら家情報を外部変数に代入してループを抜ける
      if((flg1)&&(flg3)){
        House=n_House;
        garden=n_garden;
        break;
      }else{
        // 家の位置を微妙にずらすときのずらし幅を生成
        float house_px=random(rctSizeX*2)-rctSizeX;
        float house_py=random(rctSizeY*2)-rctSizeY;
        // ずらし幅を家の位置に反映
        n_House.posX=House.posX+house_px;
        n_House.posY=House.posY+house_py;
        n_garden[0]=movePoint(new float[]{n_House.posX, n_House.posY}, (rctSizeY+gdnSizeY)/2, n_House.dirA);
      }
    }
    // この時点で家の位置が決まっていたらループを抜ける
    if((flg1)&&(flg2)&&(flg3))break;
  }
  m++;
  // この時点で家の位置が決まっていたらループを抜ける
  if((flg1)&&(flg2)&&(flg3))break;
}
/*
// 道路の生成
boolean flg7=true;
int flg8=0;
float distLtoP=worldLenX;
float[][] roadNode=makeRoadNode(house);
for(int i=0; i<Lines.size(); i++){
  flg7=true;
  float[][] t_road=(float[][])Lines.get(i);
  int _i=int(t_road[2][1]);
  if(_i>=0){
    float[][] t_roadNode=makeRoadNode((float[][])Shapes.get(_i));
    for(int j=0; j<3; j++){
```

```
      flg7=true;
      float[] line_bgn=roadNode[j];
      for(int l=0; l<45; l++){
        flg7=true;
        float[][] line_end=AminLtoP(t_road, line_bgn, minLtoP(t_road, line_bgn), l*1);
        for(int h=0; h<line_end.length; h++){
          flg7=true;
          float[][] n_road=new float[][]{line_bgn, line_end[h], {i, Shapes.size()}};
          if(distPtoP(n_road[0], n_road[1])<distLtoP){
            if((!crrShape(n_road, makeSquare(house)))&&(!crrShape(n_road, makeTrapezoid2(garden))){
              for(int g=0; g<Shapes.size(); g++){
                float[][] t_house=(float[][])Shapes.get(g);
                float[][] t_garden=(float[][])UniSpaces.get(g);
                if((crrShape(n_road, makeSquare(t_house)))||(crrShape(n_road, makeTrapezoid2(t_garden))){
                  flg7=false;
                }
              }
            }else{
              flg7=false;
            }
            if(flg7){
              road=n_road;
              road2=new float[][]{n_road[0], roadNode[1], {i, Shapes.size()}};
              distLtoP=distPtoP(n_road[0], n_road[1]);
              flg8=1;
            }
          }
        }
      }
      for(int l=0; l<3; l++){
        flg7=true;
        float[][] line_end=new float[][]{t_roadNode[l]};
        for(int h=0; h<line_end.length; h++){
          flg7=true;
          float[][] n_road=new float[][]{line_bgn, line_end[h], {i, Shapes.size()}};
          if(distPtoP(n_road[0], n_road[1])<distLtoP){
            if((!crrShape(n_road, makeSquare(house)))&&(!crrShape(n_road, makeTrapezoid2(garden))){
              for(int g=0; g<Shapes.size(); g++){
                float[][] t_house=(float[][])Shapes.get(g);
                float[][] t_garden=(float[][])UniSpaces.get(g);
                if((crrShape(n_road, makeSquare(t_house)))||(crrShape(n_road, makeTrapezoid2(t_garden))){
                  flg7=false;
                }
              }
            }else{
              flg7=false;
            }
            if(flg7){
              road=n_road;
              road2=new float[][]{n_road[0], roadNode[1], {i, Shapes.size()}};
              road3=new float[][]{n_road[1], t_roadNode[1], {i, _i}};
              distLtoP=distPtoP(n_road[0], n_road[1]);
              flg8=2;
            }
          }
        }
      }
    }
  }else{
    for(int h=0; h<3; h++){
      flg7=true;
      float[] line_bgn=roadNode[h];
      for(int k=0; k<45; k++){
        flg7=true;
        float[][] line_end=AminLtoP(t_road, line_bgn, minLtoP(t_road, line_bgn), k*1);
        for(int l=0; l<line_end.length; l++){
          flg7=true;
          float[][] n_road=new float[][]{line_bgn, line_end[l], {i, Shapes.size()}};
          if(distPtoP(line_bgn, line_end[l])<distLtoP){
            if((!crrShape(n_road, makeSquare(house)))&&(!crrShape(n_road, makeTrapezoid2(garden))){
              for(int j=0; j<Shapes.size(); j++){
                float[][] t_house=(float[][])Shapes.get(j);
                float[][] t_garden=(float[][])UniSpaces.get(j);
                if((crrShape(n_road, makeSquare(t_house)))||(crrShape(n_road, makeTrapezoid2(t_garden))){
                  flg7=false;
                }
              }
            }else{
              flg7=false;
            }
            if(flg7){
              road=n_road;
              road2=new float[][]{n_road[0], roadNode[1], {i, Shapes.size()}};
              distLtoP=distPtoP(n_road[0], n_road[1]);
              flg8=1;
            }
          }
        }
      }
    }
  }
}
if(flg8!=0){
  roadFlg=flg8;
  break;
}
*/
break;
```

```
}
if(addFlg){
  if(Houses.size()>0){
    for(int i=0; i<CHouseId.length; i++){
      House.setClosest(CHouseId[i], DistHouse[i]);
      for(int j=0; j<Houses.size(); j++){
        House t_House=(House)Houses.get(j);
        if(t_House.id==CHouseId[i]){
          t_House.setClosest(houseID, DistHouse[i]);
          Houses.set(j, t_House);
        }
      }
    }
  }
  House.setVoronoi();

  // 家の寿命設定
  House.life=(int)random(lifeWidth[0]*HperY,lifeWidth[1]*HperY);
  House.id=houseID;
  // 家を登録
  Houses.add(House);
  HouseIDs=(int[])append(HouseIDs, houseID);
  revHouseIDs=new int[houseID+1];
  for(int i=0; i<revHouseIDs.length; i++){
    revHouseIDs[i]=-1;
  }
  for(int i=0; i<HouseIDs.length; i++){
    revHouseIDs[HouseIDs[i]]=i;
  }
  if(Houses.size()>0){
    House.setNeighbor();
    for(int j=0; j<Houses.size(); j++){
      House t_House=(House)Houses.get(j);
      t_House.setNeighbor();
      Houses.set(j, t_House);
    }
  }
  // 庭を登録
  UniSpaces.add(garden);

  houseID++;
  if(frameAdvFlg)goRunFlg=false;
}
// 道を登録
/*
Lines.add(road);
if((roadFlg!=0)&&(distPtoP(road2[0], road2[1])>0)){
  Lines.add(road2);
}
if((roadFlg==2)&&(distPtoP(road3[0], road3[1])>0)){
  Lines.add(road3);
}
*/

// 家の寿命設定、家の寿命計算
for(int i=0; i<Houses.size(); i++){
  // 家の情報取得
  House t_House=(House)Houses.get(i);
  float[][] t_garden=(float[][])UniSpaces.get(i);
  // 家の寿命を取得
  int t_life=t_House.life;
  // 寿命の尽きた家を消す
  while(true){
    t_House=(House)Houses.get(i);
    t_garden=(float[][])UniSpaces.get(i);
    t_life=t_House.life;
    if(t_life<=0){
      for(int j=0; j<Houses.size(); j++){
        House tt_House=(House)Houses.get(j);
        tt_House.delClosest(t_House.id);
        tt_House.delNeighbor(t_House.id);
        Houses.set(j, tt_House);
      }
      HouseIDs=deleteFromIndex(HouseIDs, i);
      Houses.remove(i);
      revHouseIDs=new int[houseID+1];
      for(int j=0; j<revHouseIDs.length; j++){
        revHouseIDs[j]=-1;
      }
      for(int j=0; j<HouseIDs.length; j++){
        revHouseIDs[HouseIDs[j]]=i;
      }
      UniSpaces.remove(i);
      /*
      for(int j=0; j<Lines.size(); j++){
        float[][] t_road=(float[][])Lines.get(j);
        if(t_road[2][1]==i){
          Lines.set(j, new float[][]{t_road[0], t_road[1], {t_road[2][0], -1}});
        }else if(t_road[2][1]>i){
          Lines.set(j, new float[][]{t_road[0], t_road[1], {t_road[2][0], t_road[2][1]-1}});
        }
      }
      while(true){
        boolean flg6=true;
        for(int j=0; j<Lines.size(); j++){
          boolean flg5=true;
          float[][] t_road=(float[][])Lines.get(j);
          if(t_road[2][1]==-1){
```

```
          for(int k=0; k<Lines.size(); k++){
            float[][] tt_road=(float[][])Lines.get(k);
            if(tt_road[2][0]==j){
              flg5=false;
              break;
            }
          }
          if(flg5){
            flg6=false;
            Lines.remove(j);
            for(int k=0; k<Lines.size(); k++){
              float[][] tt_road=(float[][])Lines.get(k);
              if(tt_road[2][0]>j){
                Lines.set(k, new float[][]{tt_road[0], tt_road[1], {tt_road[2][0]-1, tt_road[2][1]}});
              }
            }
          }
        }
      }
      if(flg6)break;
    }
    */
  }else{
    t_House.life=t_House.life-1;
    Houses.set(i, t_House);
    break;
  }
 }
}

// エクスポート用の変数の初期化
if(exportFlg){
 println("Begin export");
 //beginRecord(PDF, "export/export_"+t+".pdf");
}

t++;
setMessage();

String number="00000000";
if(t<10){
 number="0000000"+t;
}else if((t>=10)&&(t<100)){
 number="000000"+t;
}else if((t>=100)&&(t<1000)){
 number="00000"+t;
}else if((t>=1000)&&(t<10000)){
 number="0000"+t;
}else if((t>=10000)&&(t<100000)){
 number="000"+t;
}else if((t>=100000)&&(t<1000000)){
 number="00"+t;
}else if((t>=1000000)&&(t<1000000)){
 number="0"+t;
}
if(export){
 //forPrint();
 Display(pgFP, 5, true, false);
 pgFP.save("export/slide/export"+number+".PNG");
 areaRate.print(t/HperY);
 areaRate.print(TAB);
 areaRate.print(Houses.size());
 areaRate.print(TAB);
 areaRate.print(shapeArea(Area)/Houses.size());
 areaRate.print(TAB);
 areaRate.print(rctSizeX*rctSizeY+(gdnSizeX+tan(gdnAng)*gdnSizeY*2)*gdnSizeY/2);
 areaRate.println();

 int[] gdnTypeCount=new int[]{0, 0, 0, 0, 0, 0, 0};
 if(UniSpaces.size()>0){
  for(int i=0; i<UniSpaces.size(); i++){
   float[][] t_garden=(float[][])UniSpaces.get(i);
   gdnTypeCount[int(t_garden[2][1])]=gdnTypeCount[int(t_garden[2][1])]+1;
  }
 }

 for(int i=0; i<7; i++){
  gdnType.print(gdnTypeCount[i]);
  gdnType.print(TAB);
 }
 gdnType.println();

 for(int i=0; i<appVoid.length; i++){
  float[] appVoidresult=House.appHouseVoid(100, i);
  for(int j=0; j<appVoidresult.length; j++){
   appVoid[i].print(appVoidresult[j]);
   appVoid[i].print(TAB);
  }
  appVoid[i].println();
 }

 for(int i=0; i<Houses.size(); i++){
  House t_House=(House)Houses.get(i);
  sunshine.print(t_House.appHouseSunshine());
  sunshine.print(TAB);

  vArea.print(t_House.getVoronoiArea());
  vArea.print(TAB);
```

```
          }

          sunshine.println();
          if(t<100){
            switch(t){
              case 0:
                exportFlg=true;
                break;
              case 1:
                exportFlg=true;
                break;
              case 5:
                exportFlg=true;
                break;
              case 10:
                exportFlg=true;
                break;
              case 35:
                exportFlg=true;
                break;
              case 50:
                exportFlg=true;
            }
          }else{
            if(t%100==0)
              exportFlg=true;
          }
        }
        if(exportFlg){
          pdf=createGraphics(width-200, height, PDF, "export/export_"+t+".pdf");
          //forPDF(1);
          Display(pdf, 5, true, true);
          exportFlg=false;
          println("End export");
          println();
        }
      }
      count++;
    }
    //Display();
    //Display3D();
    //textDisplay();
  }
 }
}
```

ソースコード

デザイン事例 2 : Porous Torus

```
int[][] Arr2D, Arr2DABC, VArr, VArrA, VArrB, VArrC, SArr, HTotal, HPleng, VPspan;
int[] VDice, SDice, MinArr;
float rectX, rectY;
float crlength, arcAngle;
float[] RoundLen;
int vNum, sNum, stair, divRound;
int resetCount1, resetCount2;
boolean outputFLG;
PrintWriter output1, output2, output3, output4;
PrintWriter[] OUTPUT1, OUTPUT2, OUTPUT3, OUTPUT4, OUTPUT5, OUTPUT6, OUTPUT7;

int actColumn, actSlot;
int pactColumn;
int fileNum;
boolean isLoop, isChange, skipFLG, upFLG, resetFLG, backFLG;

Date date;
SimpleDateFormat sdf;
String dataString;

void setup(){
  size(1000,500);

  date=new Date();
  sdf=new SimpleDateFormat("yyyy'.'MM'.'dd'.'HHmm");
  dataString=sdf.format(date);
  fileNum=0;

  initSetting();

  actColumn=0;
  pactColumn=0;
  actSlot=0;
  resetCount1=0;
  resetCount2=0;
  isLoop=true;
  skipFLG=false;
  upFLG=false;
  backFLG=false;
  resetFLG=false;
  Arr2D=refresh(Arr2D, VArr, SArr);
}

void draw(){
  background(0);
  backFLG=false;

  if(isLoop){
    // 段配置初期化
    Arr2D=refresh(Arr2D, VArr, SArr);
    VDice=makePip(VPspan[actColumn][0], VPspan[actColumn][1], VPspan[actColumn][2]);
    SDice=makePip(HPleng[actColumn][0], HPleng[actColumn][1], HPleng[actColumn][2]);
    vNum=HTotal[actColumn][0];
    sNum=HTotal[actColumn][0];
    resetCount2=0;
    // 段が進んだ場合
    if(actColumn>pactColumn){
      resetCount1=0;
      pactColumn=actColumn;
    // 段が戻った場合
    }else{
      resetCount1++;
    }

    // 段を戻す
    if((actColumn>0)&&(resetCount1>11)){
      resetCount1=5;
      resetCount2=5;
      backFLG=true;
      skipFLG=true;
      upFLG=true;
    }
    println();
    print((actColumn+1)+" 段目試行中 ... ");

    if(skipFLG){
      skipFLG=false;
    }else{
      while(true){
        // 段を戻す
        if(actColumn>0){
          if(resetCount2>15){
            backFLG=true;
            resetCount2=0;
            //print("    配置失敗    やり直し ");
            upFLG=true;
            break;
          }else if(resetCount1>10){
            resetCount1++;
            backFLG=true;
            upFLG=true;
            break;
          }
        }
        if(resetFLG){
          resetColumn();
          //println("reset!!");
          resetFLG=false;
```

```
}

// 垂直材の配置
if(!upFLG){
 // 段で最初の配置
 if(actSlot==HTotal[actColumn][0]){
  //print("| ");
  //1 段目の場合
  if(actColumn==0){
   int _vNum=0;
   VArr[actColumn]=append(VArr[actColumn],vNum);
   _vNum=judgeFromDice(vNum, actColumn, MinArr[actColumn], VDice);
   if(_vNum==-1){
    // 配置失敗
    resetFLG=true;
    resetCount2++;
    skipFLG=true;
   }else if(_vNum==-2){
    /*resetFLG=true;
    resetCount2++;
    skipFLG=true;*/
   }else{
    // 配置成功
    VArr[actColumn]=append(VArr[actColumn],_vNum);
    vNum=vNum+_vNum;
   }
  //1 段目以外で 1 つ目のスロットの場合
  }else if(actSlot==0){
   int _vNum=judgeFromDice(vNum, actColumn, MinArr[actColumn], VDice);
   if(_vNum==-1){
    // 配置失敗
    resetFLG=true;
    resetCount2++;
    skipFLG=true;
   }else if(_vNum==-2){
    /*resetFLG=true;
    resetCount2++;
    skipFLG=true;*/
   }else{
    // 配置成功
    VArr[actColumn]=append(VArr[actColumn],_vNum);
    vNum=vNum+_vNum;
   }
  //1 段目以外で 1 つ目以外のスロットの場合
  }else{
   //1 段下の同じスロットが空白の場合
   if(Arr2D[actColumn-1][actSlot]==0){
    vNum=vNum-VPspan[actColumn][0]/2;
    //VArr[actColumn]=append(VArr[actColumn],vNum);
    int _vNum=judgeFromDice(vNum, actColumn, MinArr[actColumn], VDice);
    if(_vNum==-1){
     // 配置失敗
     resetFLG=true;
     resetCount2++;
     skipFLG=true;
    }else if(_vNum==-2){
     /*resetFLG=true;
     resetCount2++;
     skipFLG=true;*/
    }else{
     // 配置成功
     vNum=vNum+_vNum;
     VArr[actColumn]=append(VArr[actColumn],vNum);
    }
   //1 段下の次スロットが空白の場合
   }else if(Arr2D[actColumn-1][actSlot+1]==0){
    // 配置成功
    int _vNum=1;
    VArr[actColumn]=append(VArr[actColumn],vNum+1);
    vNum=vNum+_vNum;
   // それ以外
   }else{
    // 配置失敗
    resetFLG=true;
    resetCount2++;
    skipFLG=true;
   }
  }
 // 段で最初以外の配置
 }else if((actSlot==vNum)&&(actSlot<HTotal[actColumn][1])){
  int _vNum=judgeFromDice(vNum, actColumn, MinArr[actColumn], VDice);
  if(_vNum==-1){
   // 配置失敗
   resetFLG=true;
   resetCount2++;
   skipFLG=true;
  }else if(_vNum==-2){
   /*resetFLG=true;
   resetCount2++;
   skipFLG=true;*/
  }else{
   // 配置成功
   vNum=vNum+_vNum;
   if(vNum<=HTotal[actColumn][1]){
    VArr[actColumn]=append(VArr[actColumn],_vNum);
   }
  }
 }
```

```
        // 水平分割線の配置
        if(!upFLG){
          if(skipFLG){
            skipFLG=false;
          }else{
            Arr2D=refresh(Arr2D, VArr, SArr);
            // 段で最初の配置
            if(actSlot==HTotal[actColumn][0]){
              int _sNum=0;
              //SArr[actColumn]=append(SArr[actColumn],sNum-1);
              _sNum=judgeFromDice2(sNum-1, actColumn, MinArr[actColumn], SDice);
              if(_sNum==-1){
                // 配置失敗
                resetFLG=true;
                resetCount2++;
                skipFLG=true;
              }else if(_sNum==-2){
                resetFLG=true;
                resetCount2++;
                skipFLG=true;
              }else{
                // 配置成功
                //sNum=sNum-int(random(HPleng[actColumn][0]))+1;
                //sNum=sNum+_sNum;
                //SArr[actColumn]=append(SArr[actColumn],sNum);
                sNum=sNum+_sNum;
                SArr[actColumn]=append(SArr[actColumn], sNum);
              }
            // 段で最初以外の配置
            }else if((actSlot==sNum)&&(actSlot<HTotal[actColumn][1])){
              int _sNum=judgeFromDice2(sNum, actColumn, MinArr[actColumn], SDice);
              if(_sNum==-1){
                // 配置失敗
                resetFLG=true;
                resetCount2++;
                skipFLG=true;
              }else if(_sNum==-2){
                resetFLG=true;
                resetCount2++;
                skipFLG=true;
              }else{
                // 配置成功
                if(sNum+_sNum==HTotal[actColumn][1]){
                  resetFLG=true;
                  resetCount2++;
                  skipFLG=true;
                }else{
                  sNum=sNum+_sNum;
                  if(sNum<HTotal[actColumn][1]){
                    SArr[actColumn]=append(SArr[actColumn],_sNum);
                  }
                }
              }
            }
          }
        }
      }
      Arr2D=refresh(Arr2D, VArr, SArr);
      actSlot++;
      //print("["+actColumn+","+actSlot+"]");
      if(actSlot>=Arr2D[actColumn].length){
        upFLG=true;
        resetFLG=false;
        break;
      }
    }
  }
}

if(!isLoop){
  if(outputFLG){
    for(int i=0; i<VArr.length; i++){
      int spanA=0;
      int spanB=0;
      int spanC=0;
      int spanCt=0;
      // 出目の平均値
      spanCt=VPspan[i][0]*2+(VPspan[i][1]-1)*VPspan[i][2];
      for(int j=0; j<VArr[i].length; j++){
        int spanAv=0;
        if(j==0){
          spanAv=VArr[i][j+1]*2;
        }else if(j==VArr[i].length-1){
          spanAv=VArr[i][j]*2;
        }else{
          spanAv=VArr[i][j+1]+VArr[i][j];
        }
        spanA=spanA+VArr[i][j];
        spanB=spanB+VArr[i][j];
        spanC=spanC+VArr[i][j];
        if(spanAv<spanCt-VArr[i][1]/6){
          VArrA[i]=append(VArrA[i], spanA);
          spanA=0;
        }else if(spanAv>spanCt+VArr[i][1]/6){
          VArrC[i]=append(VArrC[i], spanC);
          spanC=0;
        }else{
```

```
          VArrB[i]=append(VArrB[i], spanB);
          spanB=0;
        }
      }
    }

    int[][] countPanel=new int[stair][3];
    for(int i=0;i<countPanel.length;i++){
      countPanel[i][0]=VArrA[i].length;
      countPanel[i][1]=VArrB[i].length;
      countPanel[i][2]=VArrC[i].length;
    }

    Arr2D=refresh(Arr2D, VArr, SArr);
    write(output1, Arr2D);
    write(output2, VArr);
    write(output3, SArr);
    write(output4, countPanel);
    write(OUTPUT1, Arr2D);
    write(OUTPUT2, VArr);
    write(OUTPUT3, SArr);
    write(OUTPUT5, VArrA);
    write(OUTPUT6, VArrB);
    write(OUTPUT7, VArrC);
    outputFLG=false;
    println();
    println("printed!!");
  }
 }

 for(int i=0; i<Arr2D.length; i++){
  for(int j=0; j<Arr2D[i].length; j++){
    circleRect(i, j, Arr2D);
  }
 }
 if(isLoop){
  if(backFLG)backColumn();
  if(upFLG){
    actColumn++;
    actSlot=0;
    skipFLG=false;
    upFLG=false;
    if(actColumn>=Arr2D.length)isLoop=false;
    if(outputFLG){
      fileNum++;
      save("export/"+dataString+"/slide/export"+nf(fileNum, 6)+".PNG");
    }
  }
 }
}

int randDice(int[] _pip){
 int rand=int(random(_pip.length-0.9)+0.8);
 return _pip[rand];
}

int maxDice(int[] _pip){
 return max(_pip);
}

int[] makePip(int _bgnum, int _var, int _spn){
 int[] _pip=new int[_var];
 //_pip[0]=1;
 for(int i=0;i<_pip.length; i++){
   _pip[i]=_bgnum+_spn*i;
 }
 return _pip;
}

int judgeFromDice(int _num, int _i, int[] _dise){
 int count=0;
 while(true){
  int num=_num+randDice(_dise);
  if(_i==0){
    if(num<Arr2D[_i].length){
      if(Arr2D[_i][num]==0){
        return num-_num;
      }
    }
  }else{
    if(num<Arr2D[_i].length){
      if((Arr2D[_i][num]==0)&&(Arr2D[_i-1][num]!=1)){
        return num-_num;
      }
    }
  }
  for(int i=0; i<_dise.length; i++){
    if(_dise[i]==num-_num){
      if(i==0){
        _dise=subset(_dise, 1);
      }else if(i==_dise.length-1){
        _dise=subset(_dise, 0, _dise.length-1);
      }else{
        _dise=concat(subset(_dise, 0, i),subset(_dise, i+1));
      }
    }
  }
  if(_dise.length<1){
```

```
      if(num<Arr2D[_i].length){
        return -1;
      }else{
        return -2;
      }
    }
  }
}

int judgeFromDice(int _num, int _i, int _min, int[] _dise){
  int count=0;

  // サイコロを振って置けるかの判定
  while(true){
    int num=_num+randDice(_dise);
    if(_i==0){
      if(num<Arr2D[_i].length){
        if(Arr2D[_i][num]==0){
          return num-_num;
        }
      }
    }else{
      if(num<Arr2D[_i].length){
        boolean flg=true;
        for(int i=0; i<_min; i++){
          if((num-i>-1)&&(num+i<Arr2D[_i].length)){
            if((Arr2D[_i][num-i]==0)&&(Arr2D[_i][num+i]==0)&&(Arr2D[_i-1][num-i]!=1)&&(Arr2D[_i-1][num+i]!=1)){
              }else{
                flg=false;
              }
            }
          }
        if(flg){
          return num-_num;
        }
      }
    }

    // サイの目を減らす
    for(int i=0; i<_dise.length; i++){
      if(_dise[i]==num-_num){
        if(i==0){
          _dise=subset(_dise, 1);
        }else if(i==_dise.length-1){
          _dise=subset(_dise, 0, _dise.length-1);
        }else{
          _dise=concat(subset(_dise, 0, i),subset(_dise, i+1));
        }
      }
    }

    // 部材配置失敗
    if(_dise.length<1){
      if(num<Arr2D[_i].length){
        return -1;
      }else{
        return -2;
      }
    }
  }
}

int judgeFromDiceMax(int _num, int _i, int _min, int[] _dise){
  int count=0;

  // サイコロを振って置けるかの判定
  while(true){
    int num=_num+maxDice(_dise);
    if(_i==0){
      if(num<Arr2D[_i].length){
        if(Arr2D[_i][num]==0){
          return num-_num;
        }
      }
    }else{
      if(num<Arr2D[_i].length){
        boolean flg=true;
        for(int i=0; i<_min; i++){
          if((num-i>-1)&&(num+i<Arr2D[_i].length)){
            if((Arr2D[_i][num-i]==0)&&(Arr2D[_i][num+i]==0)&&(Arr2D[_i-1][num-i]!=1)&&(Arr2D[_i-1][num+i]!=1)){
              ;
              }else{
                flg=false;
              }
            }
          }
        if(flg){
          return num-_num;
        }
      }
    }

    // サイの目を減らす
    for(int i=0; i<_dise.length; i++){
      if(_dise[i]==num-_num){
        if(i==0){
          _dise=subset(_dise, 1);
        }else if(i==_dise.length-1){
```

```
          _dise=subset(_dise, 0, _dise.length-1);
        }else{
          _dise=concat(subset(_dise, 0, i),subset(_dise, i+1));
        }
      }
    }

    // 部材配置失敗
    if(_dise.length<1){
      if(num<Arr2D[_i].length){
        return -1;
      }else{
        return -2;
      }
    }
  }
}

int judgeFromDice2(int _num, int _i, int _min, int[] _dise){
  int count=0;

  // サイコロを振って置けるかの判定
  while(true){
    int num=_num+randDice(_dise);
    if(_i==0){
      if(num<Arr2D[_i].length){
        if(Arr2D[_i][num]==0){
          return num-_num;
        }
      }
    }else{
      if(num<Arr2D[_i].length){
        boolean flg=true;
        for(int i=0; i<_min; i++){
          if((num-i>-1)&&(num+i<Arr2D[_i].length)){
            if((Arr2D[_i][num-i]==0)&&(Arr2D[_i][num+i]==0)&&(Arr2D[_i-1][num-i]!=1)&&(Arr2D[_i-1][num+i]!=1)){
            }else{
              flg=false;
            }
          }
        }
        if(flg){
          return num-_num;
        }
      }
    }

    // サイの目を減らす
    for(int i=0; i<_dise.length; i++){
      if(_dise[i]==num-_num){
        if(i==0){
          _dise=subset(_dise, 1);
        }else if(i==_dise.length-1){
          _dise=subset(_dise, 0, _dise.length-1);
        }else{
          _dise=concat(subset(_dise, 0, i),subset(_dise, i+1));
        }
      }
    }

    // 部材配置失敗
    if(_dise.length<1){
      if(num<Arr2D[_i].length){
        return -1;
      }else{
        return -2;
      }
    }
  }
}

int[][] refresh(int[][] _arr, int[][] _varr, int[][] _sarr){
  int[][] _rarr=new int[_arr.length][];

  for(int i=0; i<_arr.length; i++){
    _rarr[i]=new int[_arr[i].length];
    for(int j=0; j<_arr[i].length; j++){
      _rarr[i][j]=0;
    }
  }
  for(int i=0; i<_varr.length; i++){
    int _j=0;
    if(_varr[i].length>0){
      for(int j=0; j<_varr[i].length; j++){
        _j=_j+_varr[i][j];
        _rarr[i][_j]=1;
      }
    }
  }
  for(int i=0; i<_sarr.length; i++){
    int _j=0;
    if(_sarr[i].length>0){
      for(int j=0; j<_sarr[i].length; j++){
        _j=_j+_sarr[i][j];
        if(_j<_arr[i].length){
          //print(i+" "+j+" "+_j+" ");
          _rarr[i][_j]=2;
        }
```

```
      }
    }
  }
  return _rarr;
}

void resetColumn(){
  actSlot=0;
  vNum=HTotal[actColumn][0];
  sNum=HTotal[actColumn][0];
  VArr[actColumn]=new int[0];
  SArr[actColumn]=new int[0];
}

void backColumn(){
  VArr[actColumn]=new int[0];
  SArr[actColumn]=new int[0];
  VArr[actColumn-1]=new int[0];
  SArr[actColumn-1]=new int[0];
  pactColumn=actColumn;
  actColumn=actColumn-2;
  if(actColumn<-1)actColumn=-1;
}

int[][] refresh(int[][] _arr, int[][] _varrA, int[][] _varrB, int[][] _varrC, int[][] _sarr){
  int[][] _rarr=new int[_arr.length][];

  for(int i=0; i<_arr.length; i++){
    _rarr[i]=new int[_arr[i].length];
    for(int j=0; j<_arr[i].length; j++){
      _rarr[i][j]=0;
    }
  }

  for(int i=0; i<_varrA.length; i++){
    int _j=0;
    if(_varrA[i].length>0){
      for(int j=0; j<_varrA[i].length; j++){
        _j=_j+_varrA[i][j];
        _rarr[i][_j]=2;
      }
    }
  }

  for(int i=0; i<_varrB.length; i++){
    int _j=0;
    if(_varrB[i].length>0){
      for(int j=0; j<_varrB[i].length; j++){
        _j=_j+_varrB[i][j];
        _rarr[i][_j]=3;
      }
    }
  }

  for(int i=0; i<_varrC.length; i++){
    int _j=0;
    if(_varrC[i].length>0){
      for(int j=0; j<_varrC[i].length; j++){
        _j=_j+_varrC[i][j];
        _rarr[i][_j]=4;
      }
    }
  }

  for(int i=0; i<_sarr.length; i++){
    int _j=0;
    if(_sarr[i].length>0){
      for(int j=0; j<_sarr[i].length; j++){
        _j=_j+_sarr[i][j];
        _rarr[i][_j]=9;
      }
    }
  }
  return _rarr;
}

void write(PrintWriter writer, int[][] _arr) {
  for (int i=0; i<_arr.length; i++) {
    if(_arr[i].length>0){
      for (int j=0; j<_arr[i].length; j++) {
        if (j!=0) {
          writer.print(TAB);
        }
        writer.print(_arr[i][j]);
      }
    }
    writer.println();
  }
  writer.flush();
}

void write(PrintWriter writer, String[][] _arr) {
  for (int i=0; i<_arr.length; i++) {
    if(_arr[i].length>0){
      for (int j=0; j<_arr[i].length; j++) {
        if (j!=0) {
          writer.print(TAB);
        }
```

```
        if (_arr[i][j]!=null) {
          writer.print(_arr[i][j]);
        }
      }
    }
    writer.println();
  }
  writer.flush();
}

void write(PrintWriter[] writer, int[][] _arr) {
  for (int i=0; i<_arr.length; i++) {
    for (int j=0; j<_arr[i].length; j++) {
      writer[i].println(_arr[i][j]);
    }
    writer[i].flush();
  }
}

void write(PrintWriter[] writer, String[][] _arr) {
  for (int i=0; i<_arr.length; i++) {
    for (int j=0; j<_arr[i].length; j++) {
      if (_arr[i][j]!=null) {
        writer[i].println(_arr[i][j]);
      }
    }
    writer[i].flush();
  }
}

void drawRect(int _i, int _j, int[][] _arr){
  switch(_arr[_i][_j]){
    case 0:
      noStroke();
      fill(50);
      rect(width-(rectY+_i*rectX*2+rectX),rectY+_j*rectY,rectY,rectY);
      rect(width-(rectX+rectY*2+_i*rectX*2+rectX),rectY+_j*rectY,rectX,rectY);
      break;
    case 1:
      noStroke();
      fill(50);
      rect(width-(rectY+_i*rectX*2+rectX),rectY+_j*rectY,rectY,rectY);
      noStroke();
      fill(200, 50, 50);
      rect(width-(rectX+rectY*2+_i*rectX*2+rectX),rectY+_j*rectY,rectX,rectY);
      break;
    case 2:
      noStroke();
      fill(50, 50, 200);
      rect(width-(rectY+_i*rectX*2+rectX),rectY+_j*rectY,rectY,rectY);
      noStroke();
      fill(50);
      rect(width-(rectX+rectY*2+_i*rectX*2+rectX),rectY+_j*rectY,rectX,rectY);
      break;
    case 3:
      noStroke();
      fill(50);
      rect(width-(rectY+_i*rectX*2+rectX),rectY+_j*rectY,rectY,rectY);
      noStroke();
      fill(50, 200, 50);
      rect(width-(rectX+rectY*2+_i*rectX*2+rectX),rectY+_j*rectY,rectX,rectY);
      break;
    case 4:
      noStroke();
      fill(50, 200, 50);
      rect(width-(rectY+_i*rectX*2+rectX),rectY+_j*rectY,rectY,rectY);
      noStroke();
      fill(50);
      rect(width-(rectX+rectY*2+_i*rectX*2+rectX),rectY+_j*rectY,rectX,rectY);
      break;
  }
}

void circleRect(int _i, int _j, int[][] _arr){
  switch(_arr[_i][_j]){
    case 0:
      if((HTotal[_i][0]==_j)||(HTotal[_i][1]==_j)){
        stroke(255);
      }else{
        noStroke();
      }
      fill(50);
      beginShape();
        vertex(width*2/5+sin(arcAngle*2*_j)*(36-_i)*crlength, height/2-cos(arcAngle*2*_j)*(36-_i)*crlength);
        vertex(width*2/5+sin(arcAngle*2*(_j+1))*(36-_i)*crlength, height/2-cos(arcAngle*2*(_j+1))*(36-_i)*crlength);
        vertex(width*2/5+sin(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength));
        vertex(width*2/5+sin(arcAngle*2*_j)*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*_j)*((36-_i)*crlength+crlength));
      endShape(CLOSE);
      break;
    case 1:
      if((HTotal[_i][0]==_j)||(HTotal[_i][1]==_j)){
        stroke(255);
      }else{
        noStroke();
      }
      fill(200, 50, 50);
      beginShape();
        vertex(width*2/5+sin(arcAngle*2*_j)*(36-_i)*crlength, height/2-cos(arcAngle*2*_j)*(36-_i)*crlength);
```

```
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*(36-_i)*crlength, height/2-cos(arcAngle*2*(_j+1))*(36-_i)*crlength);
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength));
      vertex(width*2/5+sin(arcAngle*2*_j)*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*_j)*((36-_i)*crlength+crlength));
    endShape(CLOSE);
    break;
  case 2:
    if((HTotal[_i][0]==_j)||(HTotal[_i][1]==_j)){
      stroke(255);
    }else{
      noStroke();
    }
    fill(50, 50, 200);
    beginShape();
      vertex(width*2/5+sin(arcAngle*2*_j)*(36-_i)*crlength, height/2-cos(arcAngle*2*_j)*(36-_i)*crlength);
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*(36-_i)*crlength, height/2-cos(arcAngle*2*(_j+1))*(36-_i)*crlength);
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength));
      vertex(width*2/5+sin(arcAngle*2*_j)*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*_j)*((36-_i)*crlength+crlength));
    endShape(CLOSE);
    break;
  /*
  case 3:
    if((HTotal[_i][0]==_j)||(HTotal[_i][1]==_j)){
      stroke(255);
    }else{
      noStroke();
    }
    fill(200, 200, 50);
    beginShape();
      vertex(width/2+sin(arcAngle*2*_j)*(36-_i)*crlength, height/2-cos(arcAngle*2*_j)*(36-_i)*crlength);
      vertex(width/2+sin(arcAngle*2*(_j+1))*(36-_i)*crlength, height/2-cos(arcAngle*2*(_j+1))*(36-_i)*crlength);
      vertex(width/2+sin(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength));
      vertex(width/2+sin(arcAngle*2*_j)*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*_j)*((36-_i)*crlength+crlength));
    endShape(CLOSE);
    break;
  case 4:
    if((HTotal[_i][0]==_j)||(HTotal[_i][1]==_j)){
      stroke(255);
    }else{
      noStroke();
    }
    fill(50, 200, 50);
    beginShape();
      vertex(width*2/5+sin(arcAngle*2*_j)*(36-_i)*crlength, height/2-cos(arcAngle*2*_j)*(36-_i)*crlength);
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*(36-_i)*crlength, height/2-cos(arcAngle*2*(_j+1))*(36-_i)*crlength);
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength));
      vertex(width*2/5+sin(arcAngle*2*_j)*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*_j)*((36-_i)*crlength+crlength));
    endShape(CLOSE);
    break;
  case 5:
    if((HTotal[_i][0]==_j)||(HTotal[_i][1]==_j)){
      stroke(255);
    }else{
      noStroke();
    }
    fill(50, 200, 50);
    beginShape();
      vertex(width*2/5+sin(arcAngle*2*_j)*(36-_i)*crlength, height/2-cos(arcAngle*2*_j)*(36-_i)*crlength);
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*(36-_i)*crlength, height/2-cos(arcAngle*2*(_j+1))*(36-_i)*crlength);
      vertex(width*2/5+sin(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*(_j+1))*((36-_i)*crlength+crlength));
      vertex(width*2/5+sin(arcAngle*2*_j)*((36-_i)*crlength+crlength), height/2-cos(arcAngle*2*_j)*((36-_i)*crlength+crlength));
    endShape(CLOSE);
    break;
  */
 }
}


void initSetting(){
 outputFLG=true;
 stair=32;
 divRound=720;
 output1 = createWriter("export/"+dataString+"/list.tsv");
 output2 = createWriter("export/"+dataString+"/vplist.tsv");
 output3 = createWriter("export/"+dataString+"/splist.tsv");
 output4 = createWriter("export/"+dataString+"/listABC.tsv");
 OUTPUT1=new PrintWriter[stair];
 OUTPUT2=new PrintWriter[stair];
 OUTPUT3=new PrintWriter[stair];
 OUTPUT4=new PrintWriter[stair];
 OUTPUT5=new PrintWriter[stair];
 OUTPUT6=new PrintWriter[stair];
 OUTPUT7=new PrintWriter[stair];
 for(int i=0; i<OUTPUT1.length; i++){
  OUTPUT1[i]=createWriter("export/"+dataString+"/list/list_"+i+".tsv");
  OUTPUT2[i]=createWriter("export/"+dataString+"/vplist/vplist_"+i+".tsv");
  OUTPUT3[i]=createWriter("export/"+dataString+"/splist/splist_"+i+".tsv");
  OUTPUT4[i]=createWriter("export/"+dataString+"/listABC/listABC_"+i+".tsv");
  OUTPUT5[i]=createWriter("export/"+dataString+"/vplistA/vplistA_"+i+".tsv");
  OUTPUT6[i]=createWriter("export/"+dataString+"/vplistB/vplistB_"+i+".tsv");
  OUTPUT7[i]=createWriter("export/"+dataString+"/vplistC/vplistC_"+i+".tsv");
 }
 rectX=15;
 rectY=2;
 arcAngle=PI/divRound;
 crlength=15;
 RoundLen=new float[]{23694.32, 24625.93, 25608.01, 26602.34,
                      27556.56, 28396.32, 28396.32, 29220.83,
                      28925.56, 28051.94, 26448.70, 22780.61,
```

```
                20744.98, 18709.35, 16673.71, 14638.06,
                12602.40, 10566.71,  8530.93,  6494.91,
                 3149.21,  2002.45,  1265.04,   782.39,
                  493.44,   367.49,   381.00,   486.72,
                  645.77,   832.75,  1030.12,  1225.22};
 HTotal=new int[][][{139, 192}, {140, 191}, {140, 191}, {141, 191},
              {142, 190}, {142, 190}, {142, 190}, {142, 190},
              {142, 191}, {141, 192}, {139, 192}, {137, 193},
              {129, 197}, {123, 199}, {116, 203}, {107, 207},
               {95, 213},  {77, 211},  {41, 219},   {0, 220},
                {0, 246},   {0, 277},   {0, 315},   {0, 720},
                {0, 720},   {0, 720},   {0, 720},   {0, 720},
                {0, 720},   {0, 720},   {0, 720},   {0, 720},
                {0, 720},   {0, 720},   {0, 720},   {0, 720},
                {0, 720},   {0, 720},   {0, 720},   {0, 720}};
 VPspan=new int[RoundLen.length][3];
 HPleng=new int[RoundLen.length][3];
 MinArr=new int[RoundLen.length];
 for(int i=0; i<RoundLen.length;i++){
  float spanLen=RoundLen[i]/(divRound);
  float spanLen2=(RoundLen[i]+900)/(divRound);
  int minSpan=int(48/spanLen2)+1;
  int maxSpan=int(300/spanLen2);
  int maxHP=int(750/spanLen2);
  println(spanLen);
  println(i+" "+(minSpan)+" "+(maxSpan-minSpan)+" "+1);
  println(i+" "+(maxHP-maxSpan+minSpan)+" "+(maxSpan-minSpan)+" "+1);
  VPspan[i]=new int[]{minSpan, maxSpan-minSpan, 1};
  HPleng[i]=new int[]{maxHP-maxSpan+minSpan+1, maxSpan-minSpan, 1};
  MinArr[i]=minSpan;
 }
 Arr2D=new int[stair][divRound];
 Arr2DABC=new int[stair][divRound];
 VArr=new int[stair][0];
 VArrA=new int[stair][0];
 VArrB=new int[stair][0];
 VArrC=new int[stair][0];
 SArr=new int[stair][0];
}
```

ソースコード

デザイン事例 3 : ドミエメロード / ファサードデザイン

```
import processing.pdf.*;

float scaleDis;
float marginX, marginY;
float textSizeDis;
float transome;
float gridX;
int fileNum;
int loopCount, count;
boolean exportFLG, loopStart, loopFLG, processShow, setEnd, geneEnd, firstLoop, wasSetFLG;
int[][][] MapData;
int[][] ConData;
int[][] ZeroArr;
int[][][] ConNum;
int[][] ConNumPart;
int[][] PartNum;
float[] GridYsets;
float[][] relaFLG;
float[][][] ConFLG;
int autoRunNum;
int fr;
int frCount;
int[] defSet;
int[][] Conlimits;
float louverPitch;
int colNum;

float conX, conY;
float margConX, margConY;
float matrix1X, matrix1Y,matrix2X, matrix2Y;
String[] colTitle1, colTitle2;
String[] rawTitle1, rawTitle2;
int[] activeButton1, activeButton2, activeButton3, activeButton4, selectedButton1, selectedButton2;
color activeButtonColor1, activeButtonColor2, selectedButtonColor;
int keyInput;
float sumGridY;
String extraFile;

boolean hiddenFLG;

Date date;
SimpleDateFormat sdf;

PGraphics pg, pgFP;
PImage b;
PFont font, fontBig;

void setup(){
  size(1000, 600);
  background(0);
  pg=createGraphics(width*11/20, height, JAVA2D);
  b=loadImage("import/baseElevation.png");
  pgFP=createGraphics(width*11/20, height, JAVA2D);
  frameRate(60);
  fr=1;
  frCount=0;
  textSizeDis=width*0.012;
  font=createFont("MS-Gothic", textSizeDis);
  fontBig=createFont("MS-Gothic", textSizeDis*1.15);
  textMode(MODEL);

  Conlimits=new int[][]{{4, 6, 0, 4, 0, 0},
                        {4, 6, 0, 4, 0, 0},
                        {2, 2, 0, 3, 0, 0},
                        {1, 2, 0, 3, 0, 0}};
  for(int i=0; i<Conlimits.length; i++){
    for(int j=0; j<Conlimits[i].length; j++){
      print(Conlimits[i][j]+" ");
    }
    println();
  }
  println();

  colTitle1=new String[]{" RC 壁 ", " 掃出窓 ", " 腰掛窓 ", " 仕切板 ", " 室外機 "};
  rawTitle1=new String[]{"Pattern1", "Pattern2", "Pattern3", "Pattern4"};
  matrix1X=width/18;
  matrix1Y=width/32;
  activeButton1=new int[]{-1, -1};
  activeButtonColor1=color(150, 150, 150);

  matrix2X=width/8.45;
  matrix2Y=width/30;
  colTitle2=new String[]{" 手摺 ", " スクリーン "};
  rawTitle2=new String[]{" ルーバー ", " ガラス "};
  activeButton2=new int[]{-1, -1};
  activeButtonColor2=color(0, 100);
  selectedButton1=new int[]{-1, -1};
  selectedButton2=new int[]{-1, -1};
  selectedButtonColor=color(80, 80, 80);
  keyInput=-1;

  activeButton3=new int[]{-1, -1};

  activeButton4=new int[]{-1, -1};

  autoRunNum=1;
  fileNum=0;
  exportFLG=true;
```

```
loopStart=false;
setEnd=false;
geneEnd=false;
processShow=true;
hiddenFLG=false;
if(processShow){
  loopFLG=false;
}else{
  loopFLG=true;
}
firstLoop=true;
wasSetFLG=false;
scaleDis=1.0/52.0;
count=0;
loopCount=0;
conX=pg.width;
conY=0;
gridX=600;
louverPitch=100;
transome=0;
GridYsets=new float[]{2920, 3020, 2820, 2920, 2920, 2920};
relaFLG=new float[][]{{100, 100, 100, 100},
                      { 50,  50,  25,  25},
                      { 25,  50,  50,  25},
                      {  0,   0, 100,  50},
                      {100,   0, 100,   0}};
MapData=importMap("import/map.tsv");
defSet=new int[0];
///*
defSet=new int[]{1, 1, 0, 0, 0, 3, 1, 0, 0, 1,
                 1, 2, 2, 1, 1, 0, 0, 3, 2, 0,
                 1, 1, 0, 0, 3, 0};
//*/
colNum=0;
sumGridY=0.;
ConFLG=new float[MapData.length][][];
ConData=new int[MapData.length][];
ZeroArr=new int[MapData.length][];
PartNum=new int[MapData.length][];
ConNum=new int[MapData.length][][];
ConNumPart=new int[MapData.length][];
for(int i=0; i<MapData.length; i++){
  if(colNum<MapData[i].length)colNum=MapData[i].length;
  sumGridY=sumGridY+GridYsets[i];
  ConFLG[i]=new float[MapData[i].length][];
  ConData[i]=new int[MapData[i].length];
  ZeroArr[i]=new int[MapData[i].length];
  ConNum[i]=new int[MapData[i].length][6];
  ConNumPart[i]=new int[MapData[i].length];
  PartNum[i]=new int[MapData[i].length];

  int partitionCount=0;
  for(int j=0; j<MapData[i].length; j++){
    ConFLG[i][j]=new float[]{100, 100, 100, 100};
    ConData[i][j]=-1;
    for(int k=0; k<4; k++){
      for(int l=0; l<MapData[i][j].length; l++){
        if(ConFLG[i][j][k]==0){
          ConFLG[i][j][k]=0;
        }else{
          if(relaFLG[MapData[i][j][l]][k]==0){
            ConFLG[i][j][k]=0;
          }else if(relaFLG[MapData[i][j][l]][k]==50){
            ConFLG[i][j][k]=ConFLG[i][j][k]/2;
          }else if(relaFLG[MapData[i][j][l]][k]==25){
            ConFLG[i][j][k]=ConFLG[i][j][k]/4;
          }
        }
      }
    }

    boolean partitionFLG=true;
    if(j>=MapData[i].length-1){
      partitionFLG=true;
    }else{
      for(int k=0; k<MapData[i][j].length; k++){
        if(MapData[i][j][k]==3){
          partitionFLG=true;
        }
      }
    }
    if(partitionFLG){
      PartNum[i][j]=partitionCount;
      int _j=j;
      partitionCount=1;
    }
    partitionFLG=false;
    partitionCount++;
  }
}
if(defSet.length>0){
  wasSetFLG=true;
  for(int i=0; i<MapData[0].length; i++){
    ConData[0][i]=defSet[i];
  }
}
marginX=(pg.width-(gridX*colNum*scaleDis))/2;
marginY=(pg.height-sumGridY*scaleDis)/2*0.872;
```

```
  margConX=pg.width*0.08;
  margConY=pg.height*0.09;

  date=new Date();
  sdf=new SimpleDateFormat("yyyy'.'MM'.'dd'.'HHmm");
  extraFile="";
}

void draw(){
  if(loopStart){
    generateData();
    if(processShow){
      loopFLG=false;
    }else{
      loopFLG=true;
      fairing();
    }
    loopStart=false;
    setEnd=true;
    loopCount=-1;
  }
  background(230);
  Display(pg);
  Display();
  if(geneEnd)println("done!");
  if(exportFLG){
    println("now saving...");
    Display(pgFP);
    pgFP.save("export/"+sdf.format(date)+"/"+extraFile+nf(fileNum, 6)+".png");
    fileNum++;
    exportFLG=false;
    count++;
    extraFile="";
    println("saved files!");
  }
  if((setEnd)&&(!geneEnd)){
    if(frCount==0){
      if(processShow){
        if(count<autoRunNum+1){
          if(loopCount==0){
            resetData(wasSetFLG);
            generateData();
          }else if(loopCount<0){
            loopCount=0;
          }
          if(!loopFLG){
            println(loopCount);
            loopFLG=fairingforDebug();
            println(loopFLG);
            if(!loopFLG){
              loopCount++;
              if(loopCount>50){
                loopCount=0;
              }
            }else{
              exportFLG=true;
              loopFLG=false;
              loopCount=0;
            }
          }else{
            resetData(wasSetFLG);
            generateData();
            loopCount=0;
          }
        }else{
          loopCount=0;
          count=0;
          loopFLG=false;
          geneEnd=true;
        }
      }else{
        if(count<autoRunNum){
          resetData(wasSetFLG);
          generateData();
          fairing();
          exportFLG=true;
        }else{
          count=0;
          loopFLG=false;
          geneEnd=true;
        }
      }
      frCount++;
    }else if(frCount>fr){
      frCount=0;
    }else{
      frCount++;
    }
  }
}


int[][][] importMap(String filename){
  String[] _RawData=loadStrings(filename);
  String[][] _Data2D=new String[_RawData.length][];
  String[][][] _Data3D=new String[_RawData.length][][];
  for(int i=0; i<_RawData.length; i++){
    if(trim(_RawData[i]).length()==0)continue;
```

```
    _Data2D[i]=split(_RawData[i], TAB);
    String[][] _tempData=new String[0][];
    for(int j=0; j<_Data2D[i].length; j++){
      if(_Data2D[i][j].length()>0){
        _tempData=(String[][])append(_tempData, split(_Data2D[i][j], ","));
        for(int k=0; k<_tempData.length; k++){
          _tempData[k]=trim(_tempData[k]);
        }
      }
    }
    _Data3D[i]=_tempData;
  }
  int[][][] _MapData=new int[_Data3D.length][][];
  for(int i=0; i<_Data3D.length; i++){
    _MapData[i]=new int[_Data3D[i].length][];
    for(int j=0; j<_Data3D[i].length; j++){
      _MapData[i][j]=new int[_Data3D[i][j].length];
      for(int k=0; k<_Data3D[i][j].length; k++){
        _MapData[i][j][k]=int(_Data3D[i][j][k]);
      }
    }
  }
  return _MapData;
}

// 整数行列の要素をインデックス番号から削除する
int[] deleteFromIndex(int[] _Arr, int _index){
  if((_Arr.length>0)&&(_Arr.length>_index)){
    if(_index==0){
      _Arr=(int[])subset(_Arr, 1);
    }else if(_index>=(_Arr.length-1)){
      _Arr=(int[])shorten(_Arr);
    }else{
      _Arr=(int[])concat((int[])subset(_Arr, 0, _index), (int[])subset(_Arr, _index+1));
    }
  }
  return _Arr;
}

// 整数行列の要素を要素から削除する
int[] deleteFromElement(int[] _Arr, int _element){
  int[] newArr=new int[0];
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length; i++){
      if(_Arr[i]!=_element){
        newArr=(int[])append(newArr, _Arr[i]);
      }
    }
  }
  return newArr;
}

// 整数行列から要素のインデックスを検索する
int getIndex(int[] _Arr, int _element){
  if(_Arr.length>0){
    for(int i=0; i<_Arr.length; i++){
      if(_Arr[i]==_element){
        return i;
      }
    }
  }
  return -1;
}


// 原点と座標間の距離取得
float distPtoP(float[] Arr){
  return dist(Arr[0], Arr[1], 0, 0);
}

// 座標間の距離取得
float distPtoP(float[] Arr1, float[] Arr2){
  return dist(Arr1[0], Arr1[1], Arr2[0], Arr2[1]);
}

// 座標間の距離取得
float distPtoP(float[][] Arr){
  return dist(Arr[0][0], Arr[0][1], Arr[1][0], Arr[1][1]);
}

// 線分と座標の距離取得
float distLtoP(float[][] Arr1, float[] Arr2){
  float dx=Arr1[1][0]-Arr1[0][0];
  float dy=Arr1[1][1]-Arr1[0][1];
  float a=dx*dx+dy*dy;
  if(a==0)return dist(Arr1[0][0] ,Arr1[0][1], Arr2[0], Arr2[1]);
  float b=dx*(Arr1[0][0]-Arr2[0])+dy*(Arr1[0][1]-Arr2[1]);
  float t=-(b/a);
  if(t<0)t=0;
  if(t>1)t=1;
  float x=t*dx+Arr1[0][0];
  float y=t*dy+Arr1[0][1];
  return dist(x, y, Arr2[0], Arr2[1]);
}

// 線分と座標の最短距離の点生成
float[] minLtoP(float[][] Arr1, float[] Arr2){
  float dx=Arr1[1][0]-Arr1[0][0];
```

```
  float dy=Arr1[1][1]-Arr1[0][1];
  float a=dx*dx+dy*dy;
  if(a==0)return Arr1[0];
  float b=dx*(Arr1[0][0]-Arr2[0])+dy*(Arr1[0][1]-Arr2[1]);
  float t=-(b/a);
  if(t<0)t=0;
  if(t>1)t=1;
  float x=t*dx+Arr1[0][0];
  float y=t*dy+Arr1[0][1];
  return new float[]{x, y};
}

// 線分と座標の最短距離の点から、ある角度分振れた場所にある点の生成
float[][] AminLtoP(float[][] Arr1, float[] Arr2, float[] Arr3, float _ang){
  float dx=Arr2[0]-Arr3[0];
  float dy=Arr2[1]-Arr3[1];
  float distPtoL=sqrt(dx*dx+dy*dy);
  float[] _point1=new float[]{Arr3[0]+dx*tan(_ang), Arr3[1]+dy*tan(_ang)};
  float[] _point2=new float[]{Arr3[0]-dx*tan(_ang), Arr3[1]-dy*tan(_ang)};
  if((distLtoP(Arr1, _point1)==0)&&(distLtoP(Arr1, _point2)==0)){
    return new float[][]{_point1, _point2};
  }else if(distLtoP(Arr1, _point1)==0){
    return new float[][]{_point1};
  }else if(distLtoP(Arr1, _point2)==0){
    return new float[][]{_point2};
  }else{
    return new float[][]{Arr3};
  }
}
// 座標の角度方向への移動
float[] movePoint(float[] Shape, float Length, float Direct){
  float shapeCenterX=Shape[0];
  float shapeCenterY=Shape[1];
  float cosS=cos(Direct)*Length;
  float sinS=sin(Direct)*Length;
  return new float[]{shapeCenterX+sinS, shapeCenterY-cosS};
}

// 図形の角度方向への移動 1
float[] moveShape(float[] Shape, float Length, float Direct){
  float shapeCenterX=Shape[0];
  float shapeCenterY=Shape[1];
  float cosS=cos(Direct)*Length;
  float sinS=sin(Direct)*Length;
  return new float[]{shapeCenterX+sinS, shapeCenterY-cosS, Shape[2], Shape[3]};
}

// 図形の角度方向への移動 2
float[][] moveShape(float[][] Shape, float Length, float Direct){
  float shapeCenterX=Shape[0][0];
  float shapeCenterY=Shape[0][1];
  float cosS=cos(Direct)*Length;
  float sinS=sin(Direct)*Length;
  return new float[][]{{shapeCenterX+sinS, shapeCenterY-cosS}, {Shape[1][0], Shape[1][1]},};
}

// 線分と座標の交点生成
float[] crrLtoPPoint(float[][] Arr1, float[] Arr2){
  if(crrLtoP(Arr1, Arr2))
    return Arr2;
  else
    return new float[0];
}

//2 直線の交点生成
float[] crrLinePoint(float[][] Arr1, float[][] Arr2){

  float _slopeX1=Arr1[1][0]-Arr1[0][0];
  float _slopeY1=Arr1[1][1]-Arr1[0][1];
  float _slopeX2=Arr2[1][0]-Arr2[0][0];
  float _slopeY2=Arr2[1][1]-Arr2[0][1];

  if((_slopeX1==0)&&(_slopeY1==0)&&(_slopeX2==0)&&(_slopeY2==0)){
    if(distPtoP(Arr1[0], Arr2[0])==0){
      return Arr1[0];

    }else{
      return new float[2];

    }
  }else if((abs(_slopeX1)<0.0005)&&(abs(_slopeY1)<0.0005)){
    return crrLtoPPoint(Arr2, Arr1[0]);

  }else if((abs(_slopeX2)<0.0005)&&(abs(_slopeY2)<0.0005)){
    return crrLtoPPoint(Arr1, Arr2[0]);

  }else if((abs(_slopeY1)<0.0005)&&(abs(_slopeX2)<0.0005)){
    return new float[]{Arr2[0][0], Arr1[0][1]};

  }else if((abs(_slopeX1)<0.0005)&&(abs(_slopeY2)<0.0005)){
    return new float[]{Arr1[0][0], Arr2[0][1]};

  }else if(abs(_slopeX1)<0.0005){
    float _slope2=_slopeY2/_slopeX2;
    float _crrPX=Arr1[0][0];
    float _crrPY=Arr2[0][1]+(Arr1[0][0]-Arr2[0][0])*_slope2;
    return new float[]{_crrPX, _crrPY};
```

```
  }else if(abs(_slopeX2)<0.0005){
    float _slope1=_slopeY1/_slopeX1;
    float _crrPX=Arr2[0][0];
    float _crrPY=Arr1[0][1]+(Arr2[0][0]-Arr1[0][0])*_slope1;
    return new float[]{_crrPX, _crrPY};

  }else if(abs(_slopeY1)<0.0005){
    float _slope2=_slopeX2/_slopeY2;
    float _crrPY=Arr1[0][1];
    float _crrPX=Arr2[0][0]+(Arr1[0][1]-Arr2[0][1])*_slope2;
    return new float[]{_crrPX, _crrPY};

  }else if(abs(_slopeY2)<0.0005){
    float _slope1=_slopeX1/_slopeY1;
    float _crrPY=Arr2[0][1];
    float _crrPX=Arr1[0][0]+(Arr2[0][1]-Arr1[0][1])*_slope1;
    return new float[]{_crrPX, _crrPY};

  }else if((abs(_slopeX1)>0.0005)&&(abs(_slopeX2)>0.0005)){
    float _slope1=_slopeY1/_slopeX1;
    float _slope2=_slopeY2/_slopeX2;
    if(_slope1-_slope2==0){
      return new float[0];

    }else{
      float _crrPX=(_slope1*Arr1[0][0]-_slope2*Arr2[0][0]+Arr2[0][1]-Arr1[0][1])/(_slope1-_slope2);
      float _crrPY=_slope1*(_crrPX-Arr1[0][0])+Arr1[0][1];
      return new float[]{_crrPX, _crrPY};

    }
  }else{
    float _slope1=_slopeX1/_slopeY1;
    float _slope2=_slopeX2/_slopeY2;
    if(_slope1-_slope2==0){
      return new float[0];

    }else{
      float _crrPY=(_slope1*Arr1[0][1]-_slope2*Arr2[0][1]+Arr2[0][0]-Arr1[0][0])/(_slope1-_slope2);
      float _crrPX=_slope1*(_crrPY-Arr1[0][1])+Arr1[0][0];
      return new float[]{_crrPX, _crrPY};

    }
  }
}

// 線分と線分の交点生成
float[] crrSegPoint(float[][] Arr1, float[][] Arr2){
  if(crrSegment(Arr1, Arr2)){
    return crrLinePoint(Arr1, Arr2);
  }else{
    return new float[0];
  }
}

// 図形のバウンディングボックス同士の交差判定
boolean crrBBox(float[][] Arr1, float[][] Arr2){
  float dx1=Arr1[0][0];
  float dy1=Arr1[0][1];
  float dx2=Arr1[1][0];
  float dy2=Arr1[1][1];
  float dx3=Arr2[0][0];
  float dy3=Arr2[0][1];
  float dx4=Arr2[1][0];
  float dy4=Arr2[1][1];

  if(dx1>=dx2){
    if(((dx1<dx3)&&(dx1<dx4))||((dx2>dx3)&&(dx2>dx4)))
      return false;
  }else{
    if(((dx2<dx3)&&(dx2<dx4))||((dx1>dx3)&&(dx1>dx4)))
      return false;
  }

  if(dy1>=dy2){
    if(((dy1<dy3)&&(dy1<dy4))||((dy2>dy3)&&(dy2>dy4)))
      return false;
  }else{
    if(((dy2<dy3)&&(dy2<dy4))||((dy1>dy3)&&(dy1>dy4)))
      return false;
  }

  return true;
}

// 線分と座標の交差判定
boolean crrLtoP(float[][] Arr1, float[] Arr2){
  if(distLtoP(Arr1, Arr2)<=0.005){
    return true;
  }else{
    return false;
  }
}

// 線分と線分の交差判定
boolean crrSegment(float[][] Arr1, float[][] Arr2){
  if(!crrBBox(Arr1, Arr2))return false;

  float dx1=Arr1[0][0];
```

```
  float dy1=Arr1[0][1];
  float dx2=Arr1[1][0];
  float dy2=Arr1[1][1];
  float dx3=Arr2[0][0];
  float dy3=Arr2[0][1];
  float dx4=Arr2[1][0];
  float dy4=Arr2[1][1];

  float set1=dx1-dx2;
  float set2=dy1-dy2;
  float set3=dx3-dx4;
  float set4=dy3-dy4;
  float set5=dx3-dx1;
  float set6=dy3-dy1;
  if((set1*set6-set2*set5)*
    (set1*(dy4-dy1)+set2*(dx1-dx4))>0)
      return false;
  if((-(set3*set6)+set4*set5)*
    (set3*(dy2-dy3)+set4*(dx3-dx2))>0)
      return false;
  return true;
}

// 図形と座標の包含判定
boolean InShape(float[] Arr1, float[][] Arr2){
  int count=0;
  for(int i=0; i<Arr2.length; i++) {
    float[] vector1={Arr2[i][0]-Arr1[0], Arr2[i][1]-Arr1[1]};
    float[] vector2={Arr2[(i+1)%Arr2.length][0]-Arr1[0], Arr2[(i+1)%Arr2.length][1]-Arr1[1]};
    float ext=vector1[0]*vector2[1]-vector1[1]*vector2[0];
    if((vector1[1]>=0)&&(vector2[1]<0)&&(ext>0)){
      count++;
    }else if((vector1[1]<0)&&(vector2[1]>=0)&&(ext<0)){
      count++;
    }
  }
  if((count%2)==1) {
    return true;
  }else{
    return false;
  }
}

// 図形と線分の交差判定
boolean crrShape(float[][] Arr1, float[] Arr2, float[] Arr3){
  for(int i=0; i<Arr1.length; i++){
    if(InShape(Arr2, Arr1))return true;
    if(InShape(Arr3, Arr1))return true;
    float dx1=Arr1[i][0]-Arr1[(i+1)%Arr1.length][0];
    float dy1=Arr1[i][1]-Arr1[(i+1)%Arr1.length][1];
    float dx2=Arr2[0]-Arr3[0];
    float dy2=Arr2[1]-Arr3[1];
    float dx3=Arr2[0]-Arr1[i][0];
    float dy3=Arr2[1]-Arr1[i][1];
    if(crrSegment(new float[][]{Arr1[i], Arr1[(i+1)%Arr1.length]}, new float[][]{Arr2, Arr3}))
      return true;
  }
  return false;
}

// 図形と図形の交差判定
boolean crrShape(float[][] Arr1, float[][] Arr2){
  for(int i=0; i<Arr1.length; i++){
    if(InShape(Arr1[i], Arr2))return true;
    for(int j=0; j<Arr2.length; j++){
      if(InShape(Arr2[j], Arr1))return true;
      float dx1=Arr1[i][0]-Arr1[(i+1)%Arr1.length][0];
      float dy1=Arr1[i][1]-Arr1[(i+1)%Arr1.length][1];
      float dx2=Arr2[j][0]-Arr2[(j+1)%Arr2.length][0];
      float dy2=Arr2[j][1]-Arr2[(j+1)%Arr2.length][1];
      float dx3=Arr2[j][0]-Arr1[i][0];
      float dy3=Arr2[j][1]-Arr1[i][1];
      if(crrSegment(new float[][]{Arr1[i], Arr1[(i+1)%Arr1.length]}, new float[][]{Arr2[j], Arr2[(j+1)%Arr2.length]}))
        return true;
    }
  }
  return false;
}

// 図形と図形の包含判定
boolean InShape(float[][] Arr1, float[][] Arr2){
  for(int i=0; i<Arr1.length; i++){
    if(!InShape(Arr1[i], Arr2))return false;
  }
  return true;
}


void keyPressed(){
  if(key=='r'){
    if(((!setEnd)&&(!geneEnd))||((setEnd)&&(geneEnd))){
      resetData(wasSetFLG);
      exportFLG=true;
      if(processShow){
        loopFLG=false;
      }else{
        loopFLG=true;
      }
```

```
      setEnd=false;
      geneEnd=false;
    }
  }else if(key=='R'){
    if(((!setEnd)&&(!geneEnd))||((setEnd)&&(geneEnd))){
      resetData(false);
      //generateData();
      exportFLG=true;
      if(processShow){
        loopFLG=false;
      }else{
        loopFLG=true;
        fairing();
      }
      setEnd=false;
      geneEnd=false;
    }
  }else if(key=='h'){
    if(hiddenFLG)hiddenFLG=false;
    else hiddenFLG=true;
  }else if(key=='s'){
    if((!setEnd)&&(!geneEnd)){
      loopStart=true;
      setEnd=true;
    }
  }else if(key=='p'){
    exportFLG=true;
    outputPDF();
  }else if(key=='P'){
    extraFile="#p";
    exportFLG=true;
    outputPDF();
  }else if((keyCode<=57)&&(keyCode>=48)){
    if((selectedButton1[0]>=0)&&(selectedButton1[1]>=0)){
      keyInput=keyCode-48;
    }else if((selectedButton2[0]>=0)&&(selectedButton2[1]>=0)){
      keyInput=keyCode-48;
    }
  }else if(keyCode==ENTER){
    if(keyInput>=0){
      if((selectedButton1[0]>=0)&&(selectedButton1[1]>=0)){
        if((selectedButton1[0]>0)&&(selectedButton1[1]>0)){
          Conlimits[selectedButton1[0]/2*2+selectedButton1[1]/2][0]=keyInput;
        }else if(selectedButton1[0]<=0){
          Conlimits[selectedButton1[1]/2][2]=keyInput;
          Conlimits[selectedButton1[1]/2+1][2]=keyInput;
        }else if(selectedButton1[1]<=0){
          Conlimits[selectedButton1[0]/2*2][1]=keyInput;
          Conlimits[selectedButton1[0]/2*2+1][1]=keyInput;
        }
      }else if((selectedButton2[0]>=0)&&(selectedButton2[1]>=0)){
        if((selectedButton2[0]>0)&&(selectedButton2[1]>0)){
          Conlimits[selectedButton2[0]/2*2+selectedButton2[1]/2][3]=keyInput;
          println(selectedButton2[0]/2*2+selectedButton2[1]/2);
        }else if(selectedButton2[0]<=0){
          Conlimits[selectedButton2[1]/2][5]=keyInput;
          Conlimits[selectedButton2[1]/2+1][5]=keyInput;
        }else if(selectedButton2[1]<=0){
          Conlimits[selectedButton2[0]/2*2][4]=keyInput;
          Conlimits[selectedButton2[0]/2*2+1][4]=keyInput;
        }
      }
      keyInput=-1;
    }
    selectedButton1=new int[]{-1, -1};
    selectedButton2=new int[]{-1, -1};
  }
}

void mouseClicked(){
  if((activeButton1[0]>=0)&&(activeButton1[1]>=0)){
    switch(int(relaFLG[activeButton1[0]][activeButton1[1]])){
      case 100:
        relaFLG[activeButton1[0]][activeButton1[1]]=50;
        break;
      case 50:
        relaFLG[activeButton1[0]][activeButton1[1]]=25;
        break;
      case 25:
        relaFLG[activeButton1[0]][activeButton1[1]]=0;
        break;
      case 0:
        relaFLG[activeButton1[0]][activeButton1[1]]=100;
        break;
    }
    resetRelation();
  }

  if((activeButton2[0]>=0)&&(activeButton2[1]>=0)){
    if((activeButton2[0]==selectedButton1[0])&&(activeButton2[1]==selectedButton1[1])){
      ;
    }else{
      keyInput=-1;
    }
    selectedButton1=activeButton2;
  }else{
    keyInput=-1;
  }
```

```
  if((activeButton2[0]<0)&&(activeButton2[1]<0)){
    selectedButton1=new int[]{-1, -1};
  }

  if((activeButton3[0]>=0)&&(activeButton3[1]>=0)){
    if((activeButton3[0]==selectedButton2[0])&&(activeButton3[1]==selectedButton2[1])){
      ;
    }else{
      keyInput=-1;
    }
    selectedButton2=activeButton3;
  }else{
    keyInput=-1;
  }

  if((activeButton3[0]<0)&&(activeButton3[1]<0)){
    selectedButton2=new int[]{-1, -1};
  }

  if((activeButton4[0]>=0)&&(activeButton4[1]>=0)){
    if((!setEnd)&&(!geneEnd)){
      ConData[activeButton4[0]][activeButton4[1]]=ConData[activeButton4[0]][activeButton4[1]]+1;
      if(ConData[activeButton4[0]][activeButton4[1]]>3){
        ConData[activeButton4[0]][activeButton4[1]]=0;
      }
    }else if((setEnd)&&(geneEnd)){
      int zeroCount=zeroCounter(ConFLG[activeButton4[0]][activeButton4[1]]);
      if(zeroCount>0){
        while(true){
          ConData[activeButton4[0]][activeButton4[1]]=ConData[activeButton4[0]][activeButton4[1]]+1;
          if(ConData[activeButton4[0]][activeButton4[1]]>3){
            ConData[activeButton4[0]][activeButton4[1]]=0;
          }
          if(ConFLG[activeButton4[0]][activeButton4[1]][ConData[activeButton4[0]][activeButton4[1]]]!=0){
            break;
          }
        }
      }else{
        ConData[activeButton4[0]][activeButton4[1]]=0;
      }
    }
  }
}


void drawShape(float[][] _Arr){
  beginShape();
    for(int i=0; i<_Arr.length; i++){
      vertex(_Arr[i][0]*scaleDis+marginX, _Arr[i][1]*scaleDis+marginY);
    }
  endShape(CLOSE);
}

void drawShape(float[][] _Arr, PGraphics _pg){
  _pg.beginShape();
    for(int i=0; i<_Arr.length; i++){
      _pg.vertex(_Arr[i][0]*scaleDis+marginX, _Arr[i][1]*scaleDis+marginY);
    }
  _pg.endShape(CLOSE);
}


int rollDise(float[] _Arr){
  if(random(1, _Arr[0]+_Arr[1]+_Arr[2]+_Arr[3])<_Arr[0]){
    return 0;
  }else{
    if(random(1, _Arr[1]+_Arr[2]+_Arr[3])<_Arr[1]){
      return 1;
    }else{
      if(random(1, _Arr[2]+_Arr[3])<_Arr[2]){
        return 2;
      }else if(_Arr[3]>0){
        return 3;
      }else{
        return -1;
      }
    }
  }
}

int rollDise(float[] _Arr1, int[] _Arr2){
  int _zeroNum=0;
  int _zeroIndex=0;
  float[] _Arr3=new float[_Arr1.length];
  for(int i=0; i<_Arr1.length; i++){
    _Arr3[i]=_Arr1[i];
  }
  for(int i=0; i<_Arr2.length; i++){
    if(_Arr2[i]>=0)_Arr3[_Arr2[i]]=0;
  }
  for(int i=0; i<_Arr3.length; i++){
    if(_Arr3[i]==0){
      _zeroNum++;
    }
  }
  if(_zeroNum>=_Arr1.length-1){
    return -1;
```

```
  }else{
    return rollDise(_Arr3);
  }
}

void resetData(boolean _flg){
  for(int i=0; i<ConData.length; i++){
    for(int j=0; j<ConData[i].length; j++){
      if(i==0){
        if(!_flg)ConData[i][j]=-1;
      }else{
        ConData[i][j]=-1;
      }
    }
  }
}

void generateData(){
  for(int i=0; i<ConData.length; i++){
    for(int j=0; j<ConData[i].length; j++){
      if(ConData[i][j]<0){
        ConData[i][j]=rollDise(ConFLG[i][j]);
      }else{
        wasSetFLG=true;
      }
    }
  }
}

void resetRelation(){
  for(int i=0; i<MapData.length; i++){
    for(int j=0; j<MapData[i].length; j++){
      ConFLG[i][j]=new float[]{100, 100, 100, 100};
      for(int k=0; k<4; k++){
        for(int l=0; l<MapData[i][j].length; l++){
          if(ConFLG[i][j][k]==0){
            ConFLG[i][j][k]=0;
          }else{
            if(relaFLG[MapData[i][j][l]][k]==0){
              ConFLG[i][j][k]=0;
            }else if(relaFLG[MapData[i][j][l]][k]==50){
              ConFLG[i][j][k]=ConFLG[i][j][k]/2;
            }else if(relaFLG[MapData[i][j][l]][k]==25){
              ConFLG[i][j][k]=ConFLG[i][j][k]/4;
            }
          }
        }
      }
    print(nf(int(ConFLG[i][j][0])/10, 2)+""+nf(int(ConFLG[i][j][1])/10, 2)+""+nf(int(ConFLG[i][j][2])/10, 2)+""+nf(int(ConFLG[i][j][3])/10, 2)+"]");
  }
  println();
  }
  println();
}

int zeroCounter(float[] _Arr){
  int zeroNum=0;
  if(_Arr[0]==0)zeroNum=zeroNum+1;
  if(_Arr[1]==0)zeroNum=zeroNum+1;
  if(_Arr[2]==0)zeroNum=zeroNum+1;
  if(_Arr[3]==0)zeroNum=zeroNum+1;
  return zeroNum;
}

void checking(){
  int colNum=0;
  for(int i=0; i<ConData.length; i++){
    if(colNum<MapData[i].length)colNum=MapData[i].length;
    for(int j=0; j<ConData[i].length; j++){
      ZeroArr[i][j]=zeroCounter(ConFLG[i][j]);
      if(j==0){
        ConNum[i][j][0]=1;
        ConNum[i][j][1]=1;
        ConNum[i][j][2]=1;
        ConNumPart[i][j]=1;
      }else{
        if(ConData[i][j]==ConData[i][j-1]){
          ConNum[i][j][0]=ConNum[i][j-1][0]+1;
          ConNum[i][j][1]=ConNum[i][j-1][1]+1;
          ConNum[i][j][2]=ConNum[i][j-1][2]+1;
        }else{
          if(ConData[i][j]/2==ConData[i][j-1]/2){
            ConNum[i][j][1]=ConNum[i][j-1][1]+1;
          }else{
            ConNum[i][j][1]=1;
          }
          if(ConData[i][j]%2==ConData[i][j-1]%2){
            ConNum[i][j][2]=ConNum[i][j-1][2]+1;
          }else{
            ConNum[i][j][2]=1;
          }
          ConNum[i][j][0]=1;
        }
        boolean partition=false;
        for(int k=0; k<MapData[i][j].length; k++){
          if(MapData[i][j][k]==3)partition=true;
        }
        if(partition){
```

```
        ConNumPart[i][j]=1;
      }else{
        if(ConData[i][j]/2==1){
          ConNumPart[i][j]=ConNumPart[i][j-1]+1;
        }else{
          ConNumPart[i][j]=ConNumPart[i][j-1];
        }
      }
    }
  }
}
for(int i=0; i<colNum; i++){
  boolean skipFLG=true;
  for(int j=0; j<ConData.length; j++){
    if(ConData[j].length>i){
      if(skipFLG){
        ConNum[j][i][3]=1;
        ConNum[j][i][4]=1;
        ConNum[j][i][5]=1;
      }else{
        if(ConData[j][i]==ConData[j-1][i]){
          ConNum[j][i][3]=ConNum[j-1][i][3]+1;
          ConNum[j][i][4]=ConNum[j-1][i][4]+1;
          ConNum[j][i][5]=ConNum[j-1][i][5]+1;
        }else{
          if(ConData[j][i]/2==ConData[j-1][i]/2){
            ConNum[j][i][4]=ConNum[j-1][i][4]+1;
          }else{
            ConNum[j][i][4]=1;
          }
          if(ConData[j][i]%2==ConData[j-1][i]%2){
            ConNum[j][i][5]=ConNum[j-1][i][5]+1;
          }else{
            ConNum[j][i][5]=1;
          }
          ConNum[j][i][3]=1;
        }
      }
      skipFLG=false;
    }
  }
}
for(int i=0; i<ConData[0].length; i++){
  ConNum[0][i][0]=0;
  ConNum[0][i][1]=0;
  ConNum[0][i][2]=0;
  ConNum[0][i][3]=0;
  ConNum[0][i][4]=0;
  ConNum[0][i][5]=0;
  ConNumPart[0][i]=0;
}
}

void fairing(){
  checking();
  int _loopCount=0;
  while(true){
    boolean _firedFLG=true;
    if(_loopCount>20){
      resetData(wasSetFLG);
      generateData();
      _loopCount=0;
    }
    for(int i=0; i<ConData.length; i++){
      for(int j=0; j<ConData[i].length; j++){
        if(ConData[i][j]<0){
          ConData[i][j]=rollDise(ConFLG[i][j]);
          _firedFLG=false;
        }

        int[] _ConstArr=Conlimits[ConData[i][j]];
        float partLim=0.5;
        partLim=0;
        if(partLim>0){
          if(PartNum[i][j]>0){
            if(float(ConNumPart[i][j])/float(PartNum[i][j])>partLim){
              int _j=j;
              while(true){
                _j=_j-1;
                if(_j>0){
                  if(PartNum[i][_j]>0){
                    break;
                  }else{
                    ConData[i][_j]=-1;
                  }
                }else{
                  ConData[i][_j]=-1;
                  break;
                }
              }
              _firedFLG=false;
            }
          }
        }

        boolean changeFLG=true;
        for(int k=0; k<_ConstArr.length; k++){
          if((changeFLG)&&(_ConstArr[k]>0)){
```

```
            if(ConNum[i][j][k]>_ConstArr[k]){
              int offsetI=0;
              int offsetJ=0;
              if(k<3){
                offsetJ=floor(random(float(ConNum[i][j][k])-0.0000001));
              }else if(k<6){
                offsetI=floor(random(float(ConNum[i][j][k])-0.0000001));
              }else{
                int _j=j;
                while(true){
                  _j=_j-1;
                  if((_j==0)||(ConData[i][_j]/2==1)){
                    offsetJ=j-_j;
                    break;
                  }
                }
              }
              if((offsetI==0)&&(offsetJ==0)){
                if(k%3==0){
                  ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]});
                }else if(k%3==1){
                  ConData[i][j]=rollDise(ConFLG[i][j], new int[]{(ConData[i][j]/2)*2, (ConData[i][j]/2)*2+1});
                }else{
                  ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]%2, ConData[i][j]%2+2});
                }
              }else{
                if(k%3==0){
                  ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]});
                  ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{ConData[i-offsetI][j-offsetJ]});
                }else if(k%3==1){
                  ConData[i][j]=rollDise(ConFLG[i][j], new int[]{(ConData[i][j]/2)*2, (ConData[i][j]/2)*2+1});
                  ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{(ConData[i][j]/2)*2, (ConData[i][j]/2)*2+1});
                }else{
                  ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]%2, ConData[i][j]%2+2});
                  ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{ConData[i][j]%2, ConData[i][j]%2+2});
                }
              }
              changeFLG=false;
              _firedFLG=false;
              //checking();
            }
          }
        }
      }
    }
    if(_firedFLG)break;
    checking();
    _loopCount++;
  }
}

boolean fairingforDebug(){
  checking();
  boolean _firedFLGD=true;
  for(int i=0; i<ConData.length; i++){
    for(int j=0; j<ConData[i].length; j++){
      if(ConData[i][j]<0){
        ConData[i][j]=rollDise(ConFLG[i][j]);
        _firedFLGD=false;
      }
      int[] _ConstArr=Conlimits[ConData[i][j]];

      float partLim=2/3;
      if(partLim>0){
        if(PartNum[i][j]>0){
          if(float(ConNumPart[i][j])/float(PartNum[i][j])>partLim){
            int _j=j;
            while(true){
              _j=_j-1;
              if(_j>0){
                if(PartNum[i][_j]>0){
                  break;
                }else{
                  ConData[i][_j]=-1;
                }
              }else{
                ConData[i][_j]=-1;
                break;
              }
            }
            _firedFLGD=false;
          }
        }
      }

      boolean changeFLGD=true;
      for(int k=0; k<_ConstArr.length; k++){
        if((changeFLGD)&&(_ConstArr[k]>0)){
          if(ConNum[i][j][k]>_ConstArr[k]){
            int offsetI=0;
            int offsetJ=0;
            if(k<3){
              offsetJ=floor(random(float(ConNum[i][j][k])-0.0000001));
            }else if(k<6){
              offsetI=floor(random(float(ConNum[i][j][k])-0.0000001));
            }else{
              int _j=j;
              while(true){
```

```
            _j=_j-1;
            if((_j==0)||(ConData[i][_j]/2==1)){
              offsetJ=j-_j;
              break;
            }
          }
        }
        if((offsetI==0)&&(offsetJ==0)){
          if(k%3==0){
            ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]});
          }else if(k%3==1){
            ConData[i][j]=rollDise(ConFLG[i][j], new int[]{(ConData[i][j]/2)*2, (ConData[i][j]/2)*2+1});
          }else{
            ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]%2, ConData[i][j]%2+2});
          }
        }else{
          if(k%3==0){
            if(random(100)>100./3.){
              ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]});
              if(random(100)>50.){
                ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{ConData[i-offsetI][j-offsetJ]});
              }
            }else{
              ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{ConData[i-offsetI][j-offsetJ]});
            }
          }else if(k%3==1){
            if(random(100)>100./3.){
              ConData[i][j]=rollDise(ConFLG[i][j], new int[]{(ConData[i][j]/2)*2, (ConData[i][j]/2)*2+1});
              if(random(100)>50.){
                ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{(ConData[i][j]/2)*2, (ConData[i][j]/2)*2+1});
              }
            }else{
              ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{(ConData[i][j]/2)*2, (ConData[i][j]/2)*2+1});
            }
          }else{
            if(random(100)>100./3.){
            ConData[i][j]=rollDise(ConFLG[i][j], new int[]{ConData[i][j]%2, ConData[i][j]%2+2});
              if(random(100)>50.){
                ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{ConData[i][j]%2, ConData[i][j]%2+2});
              }
            }else{
              ConData[i-offsetI][j-offsetJ]=rollDise(ConFLG[i-offsetI][j-offsetJ], new int[]{ConData[i][j]%2, ConData[i][j]%2+2});
            }
          }
        }
        changeFLGD=false;
        _firedFLGD=false;
        //checking();
      }
    }
  }
 }
}
 return _firedFLGD;
}

void drawGrid(){
 float gridY=0.;
 int colNum=0;
 for(int i=0; i<MapData.length+1; i++){
   stroke(255, 0, 0, 100);
   strokeWeight(0.5);
   noFill();
   line(marginX/2, marginY+gridY*scaleDis, width-marginX/2, marginY+gridY*scaleDis);
   if(i<MapData.length){
     if(colNum<MapData[i].length)colNum=MapData[i].length;
     gridY=gridY+GridYsets[i];
   }
 }
 for(int i=0; i<colNum+1; i++){
   stroke(255, 0, 0, 100);
   strokeWeight(0.5);
   noFill();
   line(marginX+gridX*i*scaleDis, marginY/2, marginX+gridX*i*scaleDis, height-marginY/2);
 }
}

void Display(){
 fill(233);
 image(pg, 0, 0);

 textFont(fontBig);
 stroke(0);
 fill(0);
 text("1. 壁部分の要素とファサード部材の組合せ ",
      conX+margConX-20,
      conY+margConY-15);
 textFont(font);

 boolean act1FLG=false;
 for(int i=0; i<colTitle1.length+1; i++){
   for(int j=0; j<rawTitle1.length+1; j++){
     stroke(0);
     strokeWeight(1);
     if((i>0)&&(j>0)){
       if(InShape(new float[]{mouseX, mouseY},
                  new float[][]{
                    {conX+margConX+matrix1X*i+width/40, conY+margConY+matrix1Y*j},
```

```
                {conX+margConX+matrix1X*(i+1)+width/40, conY+margConY+matrix1Y*j},
                {conX+margConX+matrix1X*(i+1)+width/40, conY+margConY+matrix1Y*(j+1)},
                {conX+margConX+matrix1X*i+width/40, conY+margConY+matrix1Y*(j+1)}
               })
     ){
       activeButton1=new int[]{i-1, j-1};
       act1FLG=true;
       fill(activeButtonColor1);
     }else{
       noFill();
     }
   }else{
     noFill();
   }
   if(i>0){
     rect(conX+margConX+matrix1X*i+width/40,
          conY+margConY+matrix1Y*j,
          matrix1X,
          matrix1Y);
   }else{
     rect(conX+margConX+matrix1X*i,
          conY+margConY+matrix1Y*j,
          matrix1X+width/40,
          matrix1Y);
   }
   noStroke();
   fill(0);
   String matrixText="";
   int textAbjust=0;
   if((i>0)&&(j>0)){
     textAbjust=int(matrix1Y/2);
     switch(int(relaFLG[i-1][j-1])){
       case 100:
         matrixText=" ◎ ";
         break;
       case 50:
         matrixText=" ○ ";
         break;
       case 25:
         matrixText=" △ ";
         break;
       case 0:
         matrixText=" X ";
         break;
     }
   }else if(i>0){
     matrixText=colTitle1[i-1];
   }else if(j>0){
     matrixText=rawTitle1[j-1];
   }
   text(matrixText,
        conX+margConX+matrix1X*i+textSizeDis/2+textAbjust+width/40,
        conY+margConY+matrix1Y*j+(matrix1Y+textSizeDis)/2);
   if((i==0)&&(j>0)){
     drawIcon(conX+margConX+matrix1X*i+width/100,
              conY+margConY+matrix1Y*j+width/450,
              j-1);
   }
  }
}
if(!act1FLG)activeButton1=new int[]{-1, -1};

String[][] MatrixTexts=new String[][]{
  {                  "", nf(Conlimits[0][2], 1), nf(Conlimits[1][2], 1)},
  {nf(Conlimits[0][1], 1), nf(Conlimits[0][0], 1), nf(Conlimits[1][0], 1)},
  {nf(Conlimits[2][1], 1), nf(Conlimits[2][0], 1), nf(Conlimits[3][0], 1)}
};

float matrix1Padding=height*0.38;
textFont(fontBig);
stroke(0);
fill(0);
text("2. 横方向の重複制限回数 ",
     conX+margConX-20,
     conY+margConY+matrix1Padding-15);
textFont(font);

for(int i=0; i<colTitle2.length+1; i++){
  for(int j=0; j<rawTitle2.length+1; j++){
    if((i==0)&&(j==0)){
      stroke(0);
      strokeWeight(1);
      noFill();
      rect(conX+margConX+matrix2X*i,
           conY+margConY+matrix1Padding+matrix2Y*j,
           matrix2X,
           matrix2Y);
    }else{
      if(i==0){
        if(j<colTitle2.length+1){
          noStroke();
          fill(0);
          textFont(font);
          String
          matrixText=rawTitle2[j-1];
          int textAbjust=0;
          text(matrixText,
               conX+margConX+matrix2X*i+textSizeDis/2+textAbjust,
```

```
            conY+margConY+matrix2Y*j+matrix1Padding+(matrix2Y+textSizeDis)/2);
        }
      }else if(j==0){
        if(i<rawTitle2.length+1){
          noStroke();
          fill(0);
          textFont(font);
          String matrixText=colTitle2[i-1];
          int textAbjust=0;
          text(matrixText,
              conX+margConX+matrix2X*i+textSizeDis/2+textAbjust,
              conY+margConY+matrix2Y*j+matrix1Padding+(matrix2Y+textSizeDis)/2);
        }
      }else{
        if((i<colTitle2.length+1)&&(j<rawTitle2.length+1))
          drawIcon(conX+margConX+matrix2X*i+width/100,
                conY+margConY+matrix2Y*j+matrix1Padding+width/450,
                j-1+(i-1)*2);
      }
      noStroke();
      fill(0);
      textFont(font);
      text(MatrixTexts[i][j],
          conX+margConX+matrix2X-textSizeDis+matrix2X*i,
          conY+margConY+matrix1Padding+matrix2Y/2+textSizeDis/2+matrix2Y*j);
      if((i==selectedButton1[0])&&(j==selectedButton1[1])){
        stroke(selectedButtonColor);
        strokeWeight(1);
        fill(255);
        rect(conX+margConX+matrix2X*i+matrix2X*0.60+1,
            conY+margConY+matrix1Padding+matrix2Y*j+1,
            matrix2X*0.40-2,
            matrix2Y-2);
        rect(conX+margConX+matrix2X*i+matrix2X*0.60+2,
            conY+margConY+matrix1Padding+matrix2Y*j+2,
            matrix2X*0.40-3,
            matrix2Y-3);
        if(keyInput>=0){
          noStroke();
          fill(0);
          textFont(font);
          text(nf(keyInput, 1),
              conX+margConX+matrix2X-textSizeDis+matrix2X*i+1,
              conY+margConY+matrix1Padding+matrix2Y/2+textSizeDis/2+matrix2Y*j+1);
        }
      }
      stroke(0);
      strokeWeight(1);
      noFill();
      rect(conX+margConX+matrix2X*i,
          conY+margConY+matrix1Padding+matrix2Y*j,
          matrix2X,
          matrix2Y);
    }
  }
}

boolean act2FLG=false;
for(int i=0; i<colTitle2.length+1; i++){
  for(int j=0; j<rawTitle2.length+1; j++){
    if((i==0)&&(j==0)){
    }else{
      if(InShape(new float[]{mouseX, mouseY},
              new float[][]{
                {conX+margConX+matrix2X*i, conY+margConY+matrix1Padding+matrix2Y*j},
                {conX+margConX+matrix2X*(i+1), conY+margConY+matrix1Padding+matrix2Y*j},
                {conX+margConX+matrix2X*(i+1), conY+margConY+matrix1Padding+matrix2Y*(j+1)},
                {conX+margConX+matrix2X*i, conY+margConY+matrix1Padding+matrix2Y*(j+1)}
              })
      ){
        if((i==selectedButton1[0])&&(j==selectedButton1[1])){
        }else{
          activeButton2=new int[]{i, j};
          stroke(activeButtonColor2);
          strokeWeight(4);
          noFill();
          rect(conX+margConX+matrix2X*i-1,
              conY+margConY+matrix1Padding+matrix2Y*j-1,
              matrix2X+3,
              matrix2Y+3);
        }
        act2FLG=true;
      }
    }
  }
}
if(!act2FLG)activeButton2=new int[]{-1, -1};

float matrix2Padding=height*0.64;
textFont(fontBig);
stroke(0);
fill(0);
text("3. 縦方向の重複制限回数 ",
    conX+margConX-20,
    conY+margConY+matrix2Padding-15);
textFont(font);

MatrixTexts=new String[][]{
```

```
  {                  "", nf(Conlimits[0][5], 1), nf(Conlimits[1][5], 1)},
  {nf(Conlimits[0][4], 1), nf(Conlimits[0][3], 1), nf(Conlimits[1][3], 1)},
  {nf(Conlimits[2][4], 1), nf(Conlimits[2][3], 1), nf(Conlimits[3][3], 1)}
};

for(int i=0; i<colTitle2.length+1; i++){
  for(int j=0; j<rawTitle2.length+1; j++){
    if((i==0)&&(j==0)){
      stroke(0);
      strokeWeight(1);
      noFill();
      rect(conX+margConX+matrix2X*i,
          conY+margConY+matrix2Padding+matrix2Y*j,
          matrix2X,
          matrix2Y);
    }else{
      if(i==0){
        if(j<colTitle2.length+1){
          noStroke();
          fill(0);
          textFont(font);
          String
          matrixText=rawTitle2[j-1];
          int textAbjust=0;
          text(matrixText,
              conX+margConX+matrix2X*i+textSizeDis/2+textAbjust,
              conY+margConY+matrix2Y*j+matrix2Padding+(matrix2Y+textSizeDis)/2);
        }
      }else if(j==0){
        if(i<rawTitle2.length+1){
          noStroke();
          fill(0);
          textFont(font);
          String matrixText=colTitle2[i-1];
          int textAbjust=0;
          text(matrixText,
              conX+margConX+matrix2X*i+textSizeDis/2+textAbjust,
              conY+margConY+matrix2Y*j+matrix2Padding+(matrix2Y+textSizeDis)/2);
        }
      }else{
        if((i<colTitle2.length+1)&&(j<rawTitle2.length+1))
          drawIcon(conX+margConX+matrix2X*i+width/100,
                conY+margConY+matrix2Y*j+matrix2Padding+width/450,
                j-1+(i-1)*2);
      }
      noStroke();
      fill(0);
      textFont(font);
      text(MatrixTexts[i][j],
          conX+margConX+matrix2X-textSizeDis+matrix2X*i,
          conY+margConY+matrix2Padding+matrix2Y/2+textSizeDis/2+matrix2Y*j);
      if((i==selectedButton2[0])&&(j==selectedButton2[1])){
        stroke(selectedButtonColor);
        strokeWeight(1);
        fill(255);
        rect(conX+margConX+matrix2X*i+matrix2X*0.60+1,
            conY+margConY+matrix2Padding+matrix2Y*j+1,
            matrix2X*0.40-2,
            matrix2Y-2);
        rect(conX+margConX+matrix2X*i+matrix2X*0.60+2,
            conY+margConY+matrix2Padding+matrix2Y*j+2,
            matrix2X*0.40-3,
            matrix2Y-3);
        if(keyInput>=0){
          noStroke();
          fill(0);
          textFont(font);
          text(nf(keyInput, 1),
              conX+margConX+matrix2X-textSizeDis+matrix2X*i+1,
              conY+margConY+matrix2Padding+matrix2Y/2+textSizeDis/2+matrix2Y*j+1);
        }
      }
      stroke(0);
      strokeWeight(1);
      noFill();
      rect(conX+margConX+matrix2X*i,
          conY+margConY+matrix2Padding+matrix2Y*j,
          matrix2X,
          matrix2Y);
    }
  }
}

boolean act3FLG=false;
for(int i=0; i<colTitle2.length+1; i++){
  for(int j=0; j<rawTitle2.length+1; j++){
    if((i==0)&&(j==0)){
    }else{
      if(InShape(new float[]{mouseX, mouseY},
              new float[][]{
                {conX+margConX+matrix2X*i, conY+margConY+matrix2Padding+matrix2Y*j},
                {conX+margConX+matrix2X*(i+1), conY+margConY+matrix2Padding+matrix2Y*j},
                {conX+margConX+matrix2X*(i+1), conY+margConY+matrix2Padding+matrix2Y*(j+1)},
                {conX+margConX+matrix2X*i, conY+margConY+matrix2Padding+matrix2Y*(j+1)}
              })
      ){
        if((i==selectedButton1[0])&&(j==selectedButton1[1])){
        }else{
```

```
          activeButton3=new int[]{i, j};
          stroke(activeButtonColor2);
          strokeWeight(4);
          noFill();
          rect(conX+margConX+matrix2X*i-1,
              conY+margConY+matrix2Padding+matrix2Y*j-1,
              matrix2X+3,
              matrix2Y+3);
        }
        act3FLG=true;
      }
    }
  }
}
if(!act3FLG)activeButton3=new int[]{-1, -1};

if(((!setEnd)&&(!geneEnd))||((setEnd)&&(geneEnd))){
  boolean act4FLG=false;
  float gridY=0;
  for(int i=0; i<ConData.length; i++){
    for(int j=0; j<ConData[i].length; j++){
      if((!setEnd)&&(!geneEnd)){
        if(i==0){
          float[][] ButtonShape=new float[][]{
            {transome+gridX*j      ,            transome+gridY},
            {-transome+gridX+gridX*j,            transome+gridY},
            {-transome+gridX+gridX*j, -transome+GridYsets[i]+gridY},
            {transome+gridX*j      , -transome+GridYsets[i]+gridY}
          };
          if(InShape(new float[]{mouseX, mouseY}, dispShape(ButtonShape))){
            activeButton4=new int[]{i, j};
            act4FLG=true;
            fill(255, 150, 0, 50);
            noStroke();
            drawShape(ButtonShape);
          }
        }
      }else if((setEnd)&&(geneEnd)){
        float[][] ButtonShape=new float[][]{
          {transome+gridX*j      ,            transome+gridY},
          {-transome+gridX+gridX*j,            transome+gridY},
          {-transome+gridX+gridX*j, -transome+GridYsets[i]+gridY},
          {transome+gridX*j      , -transome+GridYsets[i]+gridY}
        };
        if(InShape(new float[]{mouseX, mouseY}, dispShape(ButtonShape))){
          activeButton4=new int[]{i, j};
          act4FLG=true;
          fill(255, 150, 0, 50);
          noStroke();
          drawShape(ButtonShape);
        }
      }
    }
    gridY=gridY+GridYsets[i];
  }
  if(!act4FLG)activeButton4=new int[]{-1, -1};
}
}

void drawGrid(PGraphics _pg){
  float gridY=0.;
  for(int i=0; i<MapData.length+1; i++){
    _pg.stroke(255, 100, 0, 100);
    _pg.strokeWeight(1);
    _pg.noFill();
    _pg.line(marginX/2, marginY+gridY*scaleDis, _pg.width-marginX/2, marginY+gridY*scaleDis);
    if(i<MapData.length){
      gridY=gridY+GridYsets[i];
    }
  }
  for(int i=0; i<colNum+1; i++){
    _pg.stroke(255, 100, 0, 100);
    _pg.strokeWeight(0.5);
    _pg.noFill();
    _pg.line(marginX+gridX*i*scaleDis, marginY/2, marginX+gridX*i*scaleDis, _pg.height-marginY/2);
  }
}

void Display(PGraphics _pg){
  _pg.beginDraw();
  _pg.background(230);
  _pg.image(b, -27, -23, b.width, b.height);
  _pg.stroke(255, 100);
  _pg.fill(255, 100);
  _pg.rect(-1, -1, _pg.width+2, _pg.height+2);
  float gridY=0;
  for(int i=0; i<ConData.length; i++){
    for(int j=0; j<ConData[i].length; j++){
      float abjustTopY=150;
      float abjustBottomY=150;
      float gridYStr=0.;
      if(ConData[i][j]<0){
        _pg.noStroke();
        _pg.noFill();
      }else{
        _pg.stroke(0, 180);
        _pg.strokeWeight(0.5);
      }
```

```
      switch(ConData[i][j]){
        case 0:
          gridYStr=GridYsets[i]/2;
          abjustTopY=0;
          _pg.stroke(200, 170, 135);
          _pg.noFill();
          break;
        case 1:
          gridYStr=GridYsets[i]/2;
          abjustTopY=0;
          _pg.fill(150, 200, 255, 80);
          break;
        case 2:
          _pg.stroke(200, 170, 135);
          _pg.noFill();
          break;
        case 3:
          _pg.fill(200, 200, 255, 150);
          break;
      }
      if(ConData[i][j]%2==1){
        drawShape(new float[][]{
          {transome+gridX*j      ,        transome+gridY+gridYStr+abjustTopY},
          {-transome+gridX+gridX*j,        transome+gridY+gridYStr+abjustTopY},
          {-transome+gridX+gridX*j, -transome+GridYsets[i]+gridY-abjustBottomY},
          {transome+gridX*j      , -transome+GridYsets[i]+gridY-abjustBottomY}
        }, _pg);
      }else if(ConData[i][j]==0){
        int louverNum=int((GridYsets[i]-gridYStr-abjustBottomY)/louverPitch);
        for(int k=0; k<louverNum; k++){
          drawShape(new float[][]{
            {transome+gridX*j      , transome+gridY+(GridYsets[i]-gridYStr-abjustBottomY)*k/louverNum+gridYStr},
            {-transome+gridX+gridX*j, transome+gridY+(GridYsets[i]-gridYStr-abjustBottomY)*k/louverNum+gridYStr}
          }, _pg);
        }
        _pg.stroke(0, 180);
        _pg.noFill();
        drawShape(new float[][]{
          {transome+gridX*j      ,        transome+gridY+gridYStr+abjustTopY},
          {-transome+gridX+gridX*j,        transome+gridY+gridYStr+abjustTopY},
          {-transome+gridX+gridX*j, -transome+GridYsets[i]+gridY-abjustBottomY},
          {transome+gridX*j      , -transome+GridYsets[i]+gridY-abjustBottomY}
        }, _pg);
      }else if(ConData[i][j]==2){
        int louverNum=int(gridX/louverPitch);
        for(int k=0; k<louverNum; k++){
          drawShape(new float[][]{
            {transome+gridX*j+gridX*k/louverNum,     transome+gridY+gridYStr+abjustTopY},
            {transome+gridX*j+gridX*k/louverNum, -transome+GridYsets[i]+gridY-abjustBottomY}
          }, _pg);
        }
        _pg.stroke(0, 180);
        _pg.noFill();
        drawShape(new float[][]{
          {transome+gridX*j      ,        transome+gridY+gridYStr+abjustTopY},
          {-transome+gridX+gridX*j,        transome+gridY+gridYStr+abjustTopY},
          {-transome+gridX+gridX*j, -transome+GridYsets[i]+gridY-abjustBottomY},
          {transome+gridX*j      , -transome+GridYsets[i]+gridY-abjustBottomY}
        }, _pg);
      }
    }
    gridY=gridY+GridYsets[i];
  }
  if(hiddenFLG){
    _pg.image(b, -27, -23, b.width, b.height);
    _pg.stroke(255, 50);
    _pg.fill(255, 50);
    _pg.rect(-1, -1, _pg.width+2, _pg.height+2);
  }
  drawGrid(_pg);
  _pg.endDraw();
}

void outputPDF(){
  PGraphics pgPDF=createGraphics(width*11/20, height, PDF, "export/"+extraFile+sdf.format(date)+"/"+extraFile+nf(fileNum, 6)+".pdf");
  pgPDF.beginDraw();
  float gridY=0;
  for(int i=0; i<ConData.length; i++){
    for(int j=0; j<ConData[i].length; j++){
      float abjustTopY=150;
      float abjustBottomY=150;
      float gridYStr=0.;
      if(ConData[i][j]<0){
        pgPDF.noStroke();
        pgPDF.noFill();
      }else{
        pgPDF.stroke(0, 180);
        pgPDF.strokeWeight(0.5);
      }
      switch(ConData[i][j]){
        case 0:
          gridYStr=GridYsets[i]/2;
          abjustTopY=0;
          pgPDF.fill(200, 170, 135);
          break;
        case 1:
          gridYStr=GridYsets[i]/2;
          abjustTopY=0;
```

```
        pgPDF.fill(150, 200, 255);
        break;
      case 2:
        pgPDF.fill(200, 170, 135);
        break;
      case 3:
        pgPDF.fill(200, 200, 255, 150);
        break;
    }
    drawShape(new float[][]{
      {transome+gridX*j      ,        transome+gridY+gridYStr+abjustTopY},
      {-transome+gridX+gridX*j,        transome+gridY+gridYStr+abjustTopY},
      {-transome+gridX+gridX*j, -transome+GridYsets[i]+gridY-abjustBottomY},
      {transome+gridX*j      , -transome+GridYsets[i]+gridY-abjustBottomY}
    }, pgPDF);
  }
  gridY=gridY+GridYsets[i];
 }
 pgPDF.dispose();
 pgPDF.endDraw();
}

float[][] dispShape(float[][] _Arr){
 float[][] _OutArr=new float[_Arr.length][2];
 for(int i=0; i<_Arr.length; i++){
   _OutArr[i]=new float[]{_Arr[i][0]*scaleDis+marginX, _Arr[i][1]*scaleDis+marginY};
 }
 return _OutArr;
}

void drawIcon(float _startX, float _startY, int _pattern){
 stroke(0);
 strokeWeight(0.5);
 switch(_pattern){
   case 0:
     fill(200, 170, 135, 150);
     _startY=_startY+width/80;
     break;
   case 1:
     fill(150, 200, 255, 80);
     _startY=_startY+width/80;
     break;
   case 2:
     fill(200, 170, 135, 150);
     break;
   case 3:
     fill(200, 200, 255, 150);
     break;
 }
 rect(_startX, _startY, width/100, width/(40*(2-_pattern/2)));
}
```

ソースコード

デザイン事例 4 : 大連大和館・別館 / 内装天井ルーバー

```
import processing.pdf.*;

int[][] MapArr;
int[][] ConArr;
int[][][] ResArr;
int[][] BannedArr;
int[] PartsPattern;
color[] PartColor;
float marginX, marginY;
float spanX, spanY;
float louverWidth;
float scaleDis;
float fontSize;
PFont font;
PGraphics inputPG, outputPG, pgFP;
int recentX, recentY;
int fileNum, count;
int autoRunNum;
boolean processShow, conDataCom, resDataCom, goToNext, complete, exportFLG, outputFLG, loopFLG;

Date date;
SimpleDateFormat sdf;

void setup(){
  size(900, 600);
  scaleDis=1/20.;
  fontSize=6.;
  font=createFont("MS-Gothic", fontSize);
  inputPG=createGraphics(width/2, height, JAVA2D);
  outputPG=createGraphics(width/2, height, JAVA2D);
  pgFP=createGraphics(width/2, height, JAVA2D);
  conDataCom=false;
  resDataCom=false;
  complete=false;
  goToNext=false;
  processShow=true;
  exportFLG=false;
  outputFLG=true;
  loopFLG=true;
  fileNum=0;
  count=0;
  MapArr=new int[61][37];
  ConArr=new int[MapArr.length][MapArr[0].length];
  PartsPattern=new int[]{3, 6};
  ResArr=new int[PartsPattern[1]][0][];
  PartColor=new color[PartsPattern[1]];
  for(int i=0; i<PartColor.length; i++){
    PartColor[i]=color(255);
  }
  BannedArr=new int[][]{
    { 6, 15,  6, 16},
    { 7, 29,  7, 30},
    {16,  1, 18,  5},
    {16,  7, 16,  8},
    {16, 23, 18, 27},
    {16, 31, 16, 32},
    {18, 21, 18, 22},
    {35, 17, 35, 18},
    {37, 11, 37, 12},
    {39, 33, 39, 34},
    {54, 21, 54, 22},
    {58,  7, 58,  8},
    {59, 29, 59, 30}
  };
  spanX=125;
  spanY=300;
  louverWidth=30;
  marginX=(width/2-spanX*(MapArr.length-1)*scaleDis)/2;
  marginY=(height-spanY*(MapArr[0].length-1)*scaleDis)/2;
  resetMap();
  resetCon();
  resetRes();
  recentX=0;
  recentY=0;
  date=new Date();
  sdf=new SimpleDateFormat("yyyy'.'MM'.'dd'.'HHmm");
  autoRunNum=0;
}

void draw(){
  background(0);
  inputDisplay(inputPG);
  outputDisplay(outputPG);
  Display();
  if(outputFLG){
    if(exportFLG){
      println("now saving...");
      Display(pgFP);
      pgFP.save("export/"+sdf.format(date)+"/"+nf(fileNum, 4)+".png");
      outputPDF();
      fileNum++;
      count++;
      exportFLG=false;
      println("saved files!");
    }
    if(complete){
      if(count>autoRunNum){
        loopFLG=false;
```

```
      count=0;
    }else if(loopFLG){
      resetMap();
      resetCon();
      resetRes();
      resDataCom=false;
      conDataCom=false;
      complete=false;
      recentX=0;
      recentY=0;
    }
  }
 }
}
void setBanned(int _strI, int _strJ, int _endI, int _endJ){
 for(int i=_strI; i<_endI+1; i++){
  for(int j=_strJ; j<_endJ+1; j++){
   if(j==_strJ){
    if((i==_strI)&&(i>0)){
     MapArr[i-1][j]=1;
    }
    MapArr[i][j]=2;
   }else if(j==_endJ){
    if((i==_strI)&&(i>0)){
     MapArr[i-1][j]=1;
    }
    MapArr[i][j]=3;
   }else{
    MapArr[i][j]=4;
   }
  }
 }
}

void drawGrid(PGraphics _pg){
 for(int i=0; i<MapArr.length; i++){
  _pg.stroke(255, 0, 0, 150);
  _pg.strokeWeight(0.5);
  _pg.noFill();
  _pg.line(marginX+spanX*scaleDis*i,
        marginY/2,
        marginX+spanX*scaleDis*i,
        _pg.height-marginY/2);
  for(int j=0; j<MapArr[i].length; j++){
   if(i==0){
    _pg.stroke(0, 0, 255, 150);
    _pg.strokeWeight(0.5);
    _pg.noFill();
    _pg.line(marginX/2,
          marginY+spanY*scaleDis*j,
          _pg.width-marginX/2,
          marginY+spanY*scaleDis*j);
   }
  }
 }
}

void inputDisplay(PGraphics _pg){
 _pg.beginDraw();
 _pg.background(0);
 for(int i=0; i<BannedArr.length; i++){
  _pg.stroke(180, 180);
  _pg.strokeWeight(0.5);
  _pg.fill(150, 150);
  _pg.rect(marginX+spanX*scaleDis*BannedArr[i][0]-spanX*scaleDis/2,
        height-marginY-spanY*scaleDis*BannedArr[i][1],
        spanX*scaleDis*(BannedArr[i][2]-BannedArr[i][0])+spanX*scaleDis,
        -spanY*scaleDis*(BannedArr[i][3]-BannedArr[i][1]));
 }
 drawGrid(_pg);
 for(int i=0; i<MapArr.length; i++){
  for(int j=0; j<MapArr[i].length; j++){
   String mapText="";
   if(MapArr[i][j]==0){
    mapText=" ○ ";
   }else if(MapArr[i][j]==1){
    mapText=" X ";
   }else if(MapArr[i][j]==2){
    mapText=" ● ";
   }else if(MapArr[i][j]==3){
    mapText=" ■ ";
   }else if(MapArr[i][j]==4){
    mapText=" □ ";
   }
   _pg.noStroke();
   _pg.fill(255);
   _pg.textFont(font);
   _pg.text(mapText,
         marginX+spanX*scaleDis*i-fontSize/2,
         height-marginY-spanY*scaleDis*j+fontSize/2);
  }
 }

 _pg.endDraw();
}

void outputDisplay(PGraphics _pg){
 _pg.beginDraw();
```

```
  _pg.background(0);
  drawGrid(_pg);

  for(int i=0; i<BannedArr.length; i++){
    _pg.stroke(180);
    _pg.strokeWeight(0.5);
    _pg.fill(150);
    _pg.rect(marginX+spanX*scaleDis*BannedArr[i][0]-spanX*scaleDis/2,
          height-marginY-spanY*scaleDis*BannedArr[i][1],
          spanX*scaleDis*(BannedArr[i][2]-BannedArr[i][0])+spanX*scaleDis,
          -spanY*scaleDis*(BannedArr[i][3]-BannedArr[i][1]));
  }

  if(!conDataCom){
    conDataCom=geneCon(processShow, _pg);
  }else if(!resDataCom){
    recentX=0;
    recentY=0;
    resDataCom=geneRes();
  }

  for(int i=0; i<MapArr.length; i++){
    for(int j=0; j<MapArr[i].length; j++){
      String mapText="";
      if(MapArr[i][j]==0){
        mapText=" ○ ";
        print("   ○ ");
      }else if(MapArr[i][j]==1){
        mapText=" X ";
        print("   X ");
      }else if(MapArr[i][j]==2){
        mapText=" ● ";
        print("   ● ");
      }else if(MapArr[i][j]==3){
        mapText=" ■ ";
        print("   ■ ");
      }else if(MapArr[i][j]==4){
        mapText=" □ ";
        print("   □ ");
      }
      _pg.noStroke();
      _pg.fill(255, 50);
      _pg.textFont(font);
      _pg.text(mapText,
            marginX+spanX*scaleDis*i-fontSize/2,
            height-marginY-spanY*scaleDis*j+fontSize/2);
      String conText="";
      if(ConArr[i][j]==1){
        conText=" H ";
      }
      _pg.stroke(255);
      _pg.fill(255);
      _pg.textFont(font);
      _pg.text(conText,
            marginX+spanX*scaleDis*i-fontSize/2,
            height-marginY-spanY*scaleDis*j+fontSize/2);
    }
    println();
  }

  if(resDataCom){
    for(int i=0; i<ResArr.length; i++){
      for(int j=0; j<ResArr[i].length; j++){
        _pg.stroke(PartColor[i]);
        _pg.strokeWeight(louverWidth*scaleDis);
        _pg.noFill();
        _pg.line(marginX+spanX*scaleDis*ResArr[i][j][0],
              height-marginY-spanY*scaleDis*ResArr[i][j][1]+2,
              marginX+spanX*scaleDis*ResArr[i][j][0],
              height-marginY-spanY*scaleDis*ResArr[i][j][1]+spanY*scaleDis*(i+1)-2);
      }
    }
    if(!complete){
      complete=true;
      if(loopFLG)exportFLG=true;
    }
  }

  _pg.endDraw();
}

void Display(PGraphics _pg){
  _pg.beginDraw();
  _pg.background(0);
  drawGrid(_pg);

  for(int i=0; i<MapArr.length; i++){
    for(int j=0; j<MapArr[i].length; j++){
      String mapText="";
      if(MapArr[i][j]==0){
        mapText=" ○ ";
      }else if(MapArr[i][j]==1){
        mapText=" X ";
      }else if(MapArr[i][j]==2){
        mapText=" ● ";
      }else if(MapArr[i][j]==3){
        mapText=" ■ ";
      }else if(MapArr[i][j]==4){
```

```
        mapText=" □ ";
      }
      _pg.noStroke();
      _pg.fill(255, 50);
      _pg.textFont(font);
      _pg.text(mapText,
              marginX+spanX*scaleDis*i-fontSize/2,
              height-marginY-spanY*scaleDis*j+fontSize/2);
      String conText="";
      if(ConArr[i][j]==1){
        conText=" H ";
      }
      _pg.stroke(255);
      _pg.fill(255);
      _pg.textFont(font);
      _pg.text(conText,
              marginX+spanX*scaleDis*i-fontSize/2,
              height-marginY-spanY*scaleDis*j+fontSize/2);
    }
  }

  for(int i=0; i<ResArr.length; i++){
    for(int j=0; j<ResArr[i].length; j++){
      _pg.stroke(PartColor[i]);
      _pg.strokeWeight(louverWidth*scaleDis);
      _pg.noFill();
      _pg.line(marginX+spanX*scaleDis*ResArr[i][j][0],
              height-marginY-spanY*scaleDis*ResArr[i][j][1]+2,
              marginX+spanX*scaleDis*ResArr[i][j][0],
              height-marginY-spanY*scaleDis*ResArr[i][j][1]+spanY*scaleDis*(i+1)-2);
    }
  }

  _pg.endDraw();
}

void resetMap(){
  for(int i=0; i<MapArr.length; i++){
    for(int j=0; j<MapArr[i].length; j++){
      if(j>=MapArr[i].length-1){
        MapArr[i][j]=2;
      }else if(j==0){
        MapArr[i][j]=3;
      }else{
        MapArr[i][j]=0;
      }
    }
  }
  for(int i=0; i<BannedArr.length; i++){
    setBanned(BannedArr[i][0],
              BannedArr[i][1],
              BannedArr[i][2],
              BannedArr[i][3]);
  }
  for(int i=0; i<MapArr.length; i++){
    for(int j=0; j<MapArr[i].length; j++){
      if(MapArr[i][j]==2){
        if(j>0){
          if(MapArr[i][j-1]==0)MapArr[i][j-1]=1;
          if(j>1){
            if(MapArr[i][j-2]==0)MapArr[i][j-2]=1;
          }
        }
      }
    }
  }
}

void resetCon(){
  for(int i=0; i<MapArr.length; i++){
    for(int j=0; j<MapArr[i].length; j++){
      ConArr[i][j]=0;
    }
  }
}

void resetRes(){
  ResArr=new int[PartsPattern[1]][0][];
}

boolean geneCon(boolean _ps, PGraphics _pg){
  while(true){
    //
    if(_ps){
      _pg.stroke(255, 150, 0);
      _pg.strokeWeight(0.5);
      _pg.noFill();
      _pg.rect(marginX+spanX*scaleDis*recentX-fontSize/2-2,
              height-marginY-spanY*scaleDis*recentY+fontSize/2+2,
              fontSize+4,
              -fontSize-4);
    }
    int[] targetY=new int[]{recentY+PartsPattern[0], recentY+PartsPattern[1]};
    if(targetY[0]>MapArr[recentX].length-1){
      targetY[0]=MapArr[recentX].length-1;
    }
    if(targetY[1]>MapArr[recentX].length-1){
      targetY[1]=MapArr[recentX].length-1;
```

```
}

boolean scopeOn=true;
if(!goToNext){
  goToNext=true;
  ConArr[recentX][recentY]=1;
  if(recentX<MapArr.length-1){
    if(MapArr[recentX+1][recentY]==0){
      MapArr[recentX+1][recentY]=1;
    }
  }
  for(int i=recentY; i<targetY[0]; i++){
    if(MapArr[recentX][i]==2){
      scopeOn=false;
    }
  }
}else{
  int[] target=new int[]{0, 0};
  int[] Candidate=new int[0];
  for(int i=recentY; i<targetY[1]+1; i++){
    if(i<targetY[0]){
      if(MapArr[recentX][i]==2){
        target=new int[]{recentX, i};
        scopeOn=false;
        break;
      }
    }else{
      if(MapArr[recentX][i]==2){
        target=new int[]{recentX, i};
        break;
      }else if(MapArr[recentX][i]==0){
        Candidate=(int[])append(Candidate, i);
      }
    }
  }
  if((target[0]==0)&&(target[1]==0)){
    if(Candidate.length>0){
      target=new int[]{recentX, Candidate[floor(random(Candidate.length-0.0000001))]};
    }else{
      println("Retried!!");
      for(int j=0; j<MapArr[recentX].length; j++){
        ConArr[recentX][j]=0;
        if(MapArr[recentX][j]==1)MapArr[recentX][j]=0;
      }
      for(int j=0; j<MapArr[recentX].length; j++){
        if(MapArr[recentX][j]==2){
          if(j>0){
            if(MapArr[recentX][j-1]==0)MapArr[recentX][j-1]=1;
            if(j>1){
              if(MapArr[recentX][j-2]==0)MapArr[recentX][j-2]=1;
            }
          }
        }
      }
      if(recentX<MapArr.length-1){
        for(int j=0; j<MapArr[recentX+1].length; j++){
          if((MapArr[recentX+1][j]==2)||(MapArr[recentX+1][j]==3)){
            if(MapArr[recentX][j]==0)MapArr[recentX][j]=1;
          }
        }
      }
      if(recentX>0){
        recentX=recentX-1;
      }
      for(int j=0; j<MapArr[recentX].length; j++){
        ConArr[recentX][j]=0;
      }
      recentY=0;
      goToNext=false;
      return false;
    }
  }
  ConArr[target[0]][target[1]]=1;
  if(recentX<MapArr.length-1){
    if(MapArr[recentX+1][recentY]==0){
      MapArr[recentX+1][recentY]=1;
    }
  }
  recentX=target[0];
  recentY=target[1];
  if(recentX<MapArr.length-1){
    if(MapArr[recentX+1][recentY]==0){
      MapArr[recentX+1][recentY]=1;
    }
  }
  goToNext=false;
}

if(_ps){
  if(scopeOn){
    _pg.stroke(200, 200, 0);
    _pg.strokeWeight(0.5);
    _pg.noFill();
    _pg.rect(marginX+spanX*scaleDis*recentX-fontSize/2-2,
             height-marginY-spanY*scaleDis*targetY[0]+fontSize/2+2,
             fontSize+4,
             -spanY*scaleDis*(targetY[1]-targetY[0])-fontSize-4);
  }
```

```
    }

    if(MapArr[recentX][recentY]==2){
      if(recentY>=MapArr[recentX].length-1){
        if(recentX>=MapArr.length-1){
          return true;
        }else{
          goToNext=false;
          recentX=recentX+1;
          recentY=0;
        }
      }else{
        for(int i=recentY; i<MapArr[recentX].length; i++){
          if(MapArr[recentX][i]==3){
            goToNext=false;
            recentY=i;
            break;
          }
        }
      }
    }
    if(_ps)break;
  }
  return false;
}

boolean geneRes(){
  int partLen=0;
  for(int i=0; i<MapArr.length; i++){
    for(int j=0; j<MapArr[i].length; j++){
      if(ConArr[i][j]==1){
        if((MapArr[i][j]!=3)&&(j!=0)){
          ResArr[partLen]=(int[][])append(ResArr[partLen], new int[]{i, j});
        }
        partLen=0;
      }else{
        partLen++;
      }
    }
  }
  return true;
}

void drawGrid(){
  for(int i=0; i<MapArr.length; i++){
    stroke(255, 0, 0, 150);
    strokeWeight(0.5);
    noFill();
    line(marginX+spanX*scaleDis*i,
         marginY/2,
         marginX+spanX*scaleDis*i,
         height-marginY/2);
    for(int j=0; j<MapArr[i].length; j++){
      if(i==0){
        stroke(0, 0, 255, 150);
        strokeWeight(0.5);
        noFill();
        line(marginX/2,
             marginY+spanY*scaleDis*j,
             width-marginX/2,
             marginY+spanY*scaleDis*j);
      }
    }
  }
}

void Display(){
  image(inputPG, 0, 0);
  image(outputPG, width/2, 0);
}

void outputPDF(){
  PGraphics pgPDF=createGraphics(int(width/2/scaleDis), int(height/scaleDis), PDF, "export/"+sdf.format(date)+"/"+nf(fileNum, 4)+".pdf");
  pgPDF.beginDraw();
  for(int i=0; i<BannedArr.length; i++){
    pgPDF.stroke(0);
    pgPDF.strokeWeight(0.5);
    pgPDF.noFill();
    pgPDF.rect(marginX/scaleDis+spanX*BannedArr[i][0]-spanX/2,
           pgPDF.height-marginY/scaleDis-spanY*BannedArr[i][1],
           spanX*(BannedArr[i][2]-BannedArr[i][0])+spanX,
           -spanY*(BannedArr[i][3]-BannedArr[i][1]));
  }
  pgPDF.colorMode(HSB, 255, 255, 255, 255);
  for(int i=0; i<ResArr.length; i++){
    for(int j=0; j<ResArr[i].length; j++){
      pgPDF.noStroke();
      pgPDF.fill(255*i/ResArr.length, 255, 255);
      pgPDF.rect(marginX/scaleDis+spanX*ResArr[i][j][0]-louverWidth/2,
             pgPDF.height-marginY/scaleDis-spanY*ResArr[i][j][1]+25,
             louverWidth,
             spanY*(i+1)-50);
    }
  }
  pgPDF.colorMode(RGB, 255, 255, 255, 255);
  pgPDF.dispose();
  pgPDF.endDraw();
}
```

```
void keyPressed(){
 if(key=='r'){
   resetMap();
   resetCon();
   resetRes();
   resDataCom=false;
   conDataCom=false;
   complete=false;
   loopFLG=true;
   recentX=0;
   recentY=0;
 }else if((keyCode<=48+PartsPattern[1])&&(keyCode>=49)){
   if((conDataCom)&&(resDataCom)){
     if(PartColor[keyCode-49]==color(255)){
       PartColor[keyCode-49]=color(150, 150, 150);
     }else if(PartColor[keyCode-49]==color(150, 150, 150)){
       PartColor[keyCode-49]=color(255);
     }
   }
 }
}

void drawShape(float[][] _Arr){
 beginShape();
   for(int i=0; i<_Arr.length; i++){
     vertex(_Arr[i][0]*scaleDis+marginX, _Arr[i][1]*scaleDis+marginY);
   }
 endShape(CLOSE);
}

void drawShape(float[][] _Arr, PGraphics _pg){
 _pg.beginShape();
   for(int i=0; i<_Arr.length; i++){
     _pg.vertex(_Arr[i][0]*scaleDis+marginX, _Arr[i][1]*scaleDis+marginY);
   }
 _pg.endShape(CLOSE);
}
```