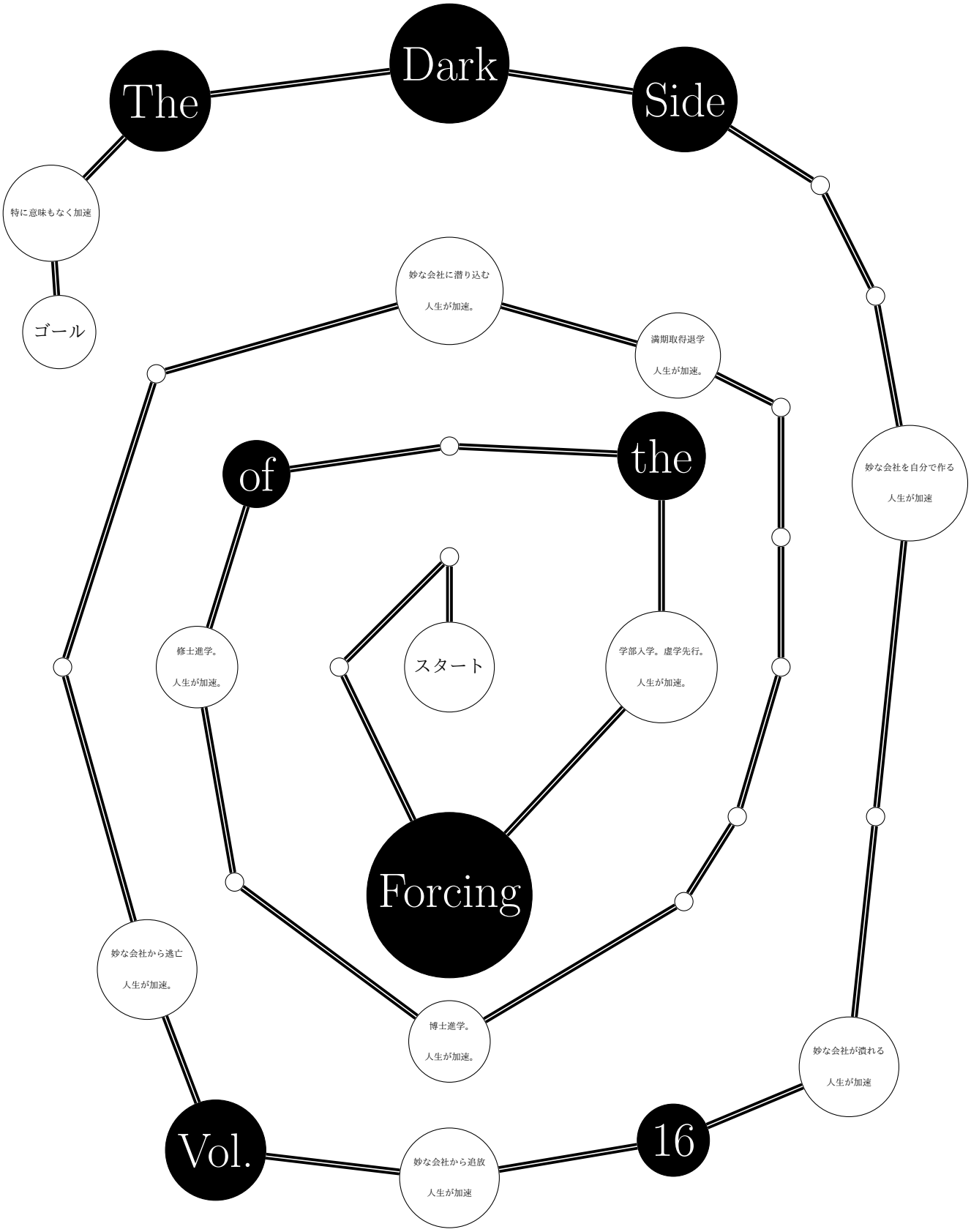


これが加速主義双六だ！！

加速主義双六ルール：

1. コマはポケットの中の埃屑とか落ちてる消しゴムなどを使う。
2. 加速したらダイスを増やす。
3. ゴールは、結婚とか仕事の成功とか、人生のゴールっぽいものを意味します。意味は人それぞれです。
4. ゴールに入った時にダイスの目の合計が余っていたら戻ります。そしてまたゴールを目指します。
5. ダイスの目の合計が「世界脱出速度＝30」を超えたら双六世界から飛び出します。それが良いことか悪いことかは人それぞれです。
6. さあ、加速しよう！

加速せよ

淡中 圏

加速せよ。
どこぞなりともこの世の外へ向けて。

地球脱出速度は地球の重力圏から

太陽系脱出速度は太陽の重力圏から

銀河系脱出速度は銀河系の重力圏から

宇宙脱出速度はこの宇宙から

人生脱出速度は人生から

目次

加速せよ i

第 1 章 数学にとって美とは何か？ プログラミングにとって美とは何か？ 1
1.1 イントロダクション 1
1.2 数学にとって美とは何か？ 1
1.3 プログラミングにとって美とは何か？ 9

参考文献 15

第 2 章 Tikz で双六を作ろう 17
2.1 イントロダクション 17
2.2 なぜ Tikz で双六を描くのか 17
2.3 Tikz の簡単な説明 18
2.4 双六ソースコード 25

宣伝小説:ズボンを買いに行くためのズボンがないので慌てて用意する話 31

第 1 章

数学にとって美とは何か？ プログラミングにとって美とは何か？

淡中 圏

1.1　イントロダクション

数学では解答や計算などについて「エレガントだ」「美しい」などの言い方をする。数学セミナーの「エレガントな解答を求む」というコーナーも有名である。

同様にプログラミングでも、ソースコードについて「エレガントかどうか」「美しいかどうか」などの判断基準がある[*1]。

ここで、

- そもそも数学にとって美とは何か？　エレガントさとは何か？
- またプログラミングにとって美とは何か？　エレガントさとは何か？
- 上記の二つは同じか？　違うとしたらどう違うか？

などの疑問が自然に生じる。

このエッセイは上の疑問に包括的な答えを与えるわけではないが、私の経験からこれらについていくつかの考えを非形式的に述べるものである。

気楽に読んでいただきたい。

1.2　数学にとって美とは何か？

このような非常に漠然とした問題について考えるとき、ぜひとも例が欲しいところである。

そこで誰でも知っている定理を持ち出してみよう。それは多角形の内閣の和の公式である。[*2]。

三角形の内角の和が 180 度であることは、おそらく読者のみなさんも覚えておいでであ

[*1] 数学ほど強く表には出てこない気はする。それはおそらくプログラミングの方が実用に重きを置いているからであろう。

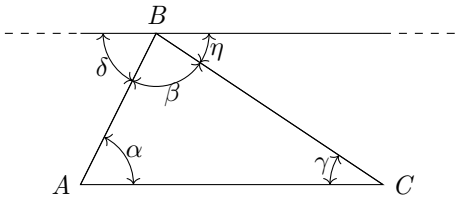
[*2] この例は志村五郎氏の『数学の好きな人のために　——続・数学をいかに使うか』[1] の第 1 章「Gauss-Bonnet の公式」から素材を得ている。

ろう[*3]。

しかしその証明は覚えていないと思うので、オーソドックスなものをここに示してみる[*4]。

定理 1 任意の三角形の三つの角の角度の総和は 180 度である。

証明　三角形 ABC の頂点 A を通り辺 BC と並行な直線が丁度一本存在する[*5]。図にすると



となる。上図の γ と δ、α と η はそれぞれ**平行線における錯覚の関係**なので等しい。よって、

$$\alpha + \beta + \gamma = \eta + \beta + \delta$$

であり、右辺のそれぞれの角を合わせたものは図より明らかに 1 直線なので、右辺は *180* 度である。

これを任意の n 角形に一般化したのが次の定理である[*6]。

定理 2 任意の n 角形の内角の和は

$$180(n-2)$$

度になる。

この定理のよくある証明は、これを三角形の場合に帰着することである。

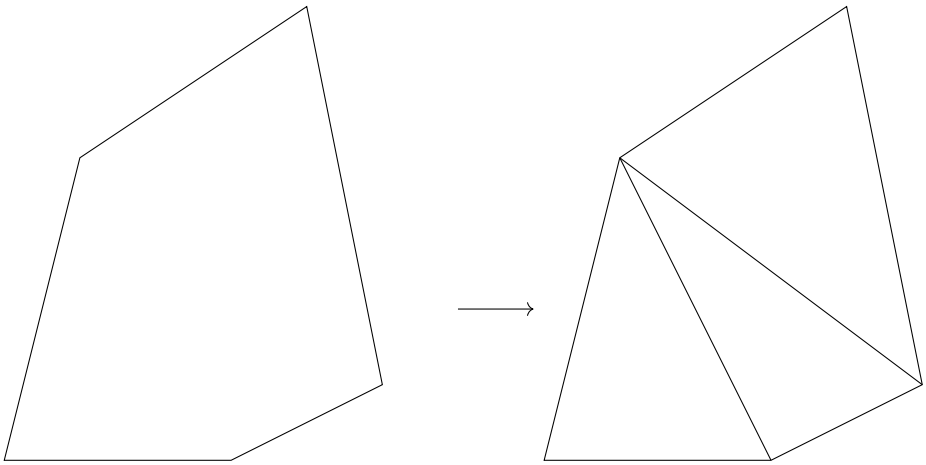
証明　n 角形の任意の頂点を一つ選び、そこから残りの頂点に向けて対角線を引く。そうすると、$n-2$ 個の三角形ができる。それぞれの三角形の内閣の和は 180 度であり、図より元の n 角形の内閣の和はこれらの三角形の内閣の和と等しくなる。

[*3] と願う。著者は高校の数学教師をしていた経験があるので、習ったもののほとんどを忘れていることは理解しているつもりだが、これくらいは覚えておいてほしい、というわずかな願いがこの程度はある。

[*4] パスカルは１０歳より前にこの定理の証明を自力でした、という逸話が残っている。

[*5] これはプレイフェアの公理と呼ばれるもので、ユークリッドの第一から第四公準を仮定すれば、有名なユークリッドの第五公準と同値な主張になる。そしてこのユークリッドの第五公準を否定する別の公理を仮定することで、非ユークリッド幾何学が産まれる。当然それらの非ユークリッド幾何学においては三角形の内閣の総和は 180 度にはならない。

[*6] 覚えているかどうかはわからないが、ここまでは義務教育で習っているはずである。



よって、元の n 角形の内閣の和は $180(n-2)$ 度となり、これは求めたかった式そのものである。

さて、ここで次の質問について考えてみよう。

この証明に不満はないか？

普通はないのではないかと思う。というより「証明への不満」という感覚について理解することが難しいと思う。

このエッセイで説明したいのは、まさにその感覚で、実際少し深く考えてみるとこの証明にはいくつかの不満点を述べることができる。そうすることで涵養されるのが数学に対する美的感覚であり、その美的感覚の正体について少し考えてみよう、というのがこのエッセイの目的についてのもう少し深い説明である。

それでは早速上記の証明への不満点であるが、第一に $n=3$ の場合が特別扱いされていることである。できるだけ全ての場合を同等に扱う方がエレガントである、というのが数学全般に対する、なんとなくの合意であろうと思われる。

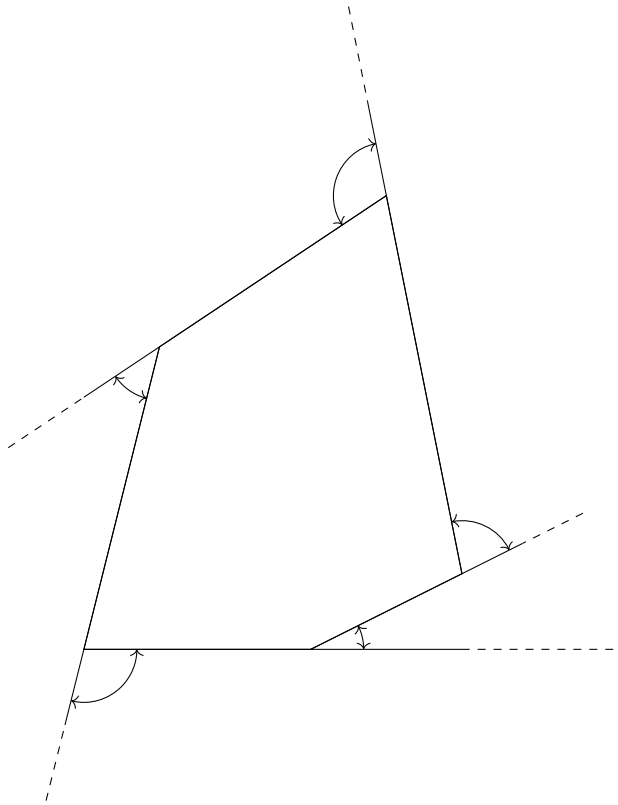
これに対して「3 は n 角形の n として最小の数なのだから、特別扱いしてもそれほど不自然ではないだろう」という突っ込みがありえて、そしてそれはとても正当な指摘であると私も首肯する。だからこれは先程の証明の瑕疵とは言うのはかなり難しい。先程の証明が不自然だとは、よほどのこだわりでも持っていなければ言えないであろう。

ただそれでも $n=3$ を特別扱いしない方法はないだろうか、と考えるとそれは実際にある。

それを説明するために次の定理に注目する。

定理 3 任意の n 角形の外角の角度の和は 360 度になる。

ここで外角とは次の図に示されるような角のことである。



外角の取り方が角ごとに 2 種類あることに気づいたかもしれないが、その二つの候補はお互いに対頂角の関係にあるので、角度は等しい。

この定理は通常は、先程の内角に関する定理から導かれる。

証明　外角には必ず対応する内角が存在し、二つの和は 180 である。よって、内閣の和を I、外角の和を O としたとき、

$$I + 0 = n * 180$$

が成立する。よって、先程の定理より

$$O = n * 180 - I = n * 180 - (n - 2) * 180 = 2 * 180 = 360$$

が成り立つ

ところでこの定理は、内角に関する定理よりもある意味で「美しい」。

なぜならこの定理は多角形の「不変量」に関しての主張だからだ。

内角に関する定理も全ての多角形に関しての情報を述べているが、この場合は、全ての多角形に関して全く同じになる量を取り出している。

不変量の考え方が物理学においてどれだけ重要かはさまざまな文献に書いてあるだろうが、ここで再び強調したい。

人間が何かを理解するということにとって、ある変化の系列の中で不変なものを探すことが非常に重要である。

心理学においては、赤子は 4 までの数字を数えることができると言われている。その実験方法は、

1. 3 つのボールを被験者に見せる。

2. そのボールを衝立で隠す。
3. そこに新しいボールを加える。
4. 衝立を取ってボールを見せる。
5. ボールの数が 4 つの場合とそれ以外の場合で反応を比べる。

というものである。この実験をすると、ボールの数が 4 つでない場合、赤子は統計的に有意に「見つめる時間」が長くなるという。これによって、赤子は物体の数を 4 つまで数えることができると結論づけられる。

私がここで言いたいのは、赤子のうちから我々は「不変量」を探している、ということである。我々は「変わらないものは何か」という質問に答えることによって、ほとんどのものが変わってしまうこの世の中で、何かを判断して生きていくことができるのではなかろうか。

これに関して私がもう一つ思い出すのは、ギリシャ哲学者のターレスである。彼はこの世界の最も基本となる構成要素「アルケー」は「水」だと主張したことで有名で、この科学的とはとても思えない言説だけ読むと「古代の人は呑気だなあ」としか思えないのだが、そもそも「世界の最も基本となる構成要素は何か？」という質問を出したのがとにかく偉い、と私は考えている。これはまさに「変わらないものは何か」という質問を人類が意識的にし始めた初期の例と言えよう[*7]。そうすると、彼に「長さの分かっている棒の影の長さを測ることで、ピラミッドの高さを計測した」という逸話があるのも非常に興味深くなる。こここでは「物体の高さと影の長さの比が同時刻では変わらない」という意味で「不変量」を使った議論になっている[*8]。私には、アルケーに関する話とこの影の長さに関する話が「変わらないものは何か」という共通の質問から駆動されているように見えて仕方ない（これはとても実証できる話ではなく、単なる私の感想に過ぎないのだが)。

そしてこのような視点を獲得しないと、例えば初等物理学における**仕事**の概念もなかなか理解できないのではないかと私は考える。なぜ、

$$\text{仕事} = \text{力} \times \text{距離}$$

などという、不思議な量を考えるかというと、単純に「てこの原理などにおいて、力と距離の積が一定になる」からである。さまざまに変化させたときに一定になるのだから、おそらく大事な量なのであろう、という単純な推論によって、**仕事**という概念は世界から選び出されている。

これは、これからの人類の歴史においてもそうそう変わらないことの一つのように私には思えるのだ。

だからこそ、私には内角よりも外角の方が、概念としての「筋」が良いように思える。そして、その筋が良い外角の方の証明を、内角を経由せずに証明したい、という欲望が芽生えるのだ。

そして、外角の和についての定理を直接証明しようとすると、それは明らかによりエレガントなものになる。

[*7] そういう意味で、不変量を探すことは科学以前の営為であり、不変量を探してるからと言って、科学的とは言えない。

[*8] 伝説上ピタゴラスがターレスに教えを受けたことになっていることからも、この手の数学的な議論の祖として考えられていたのであろうか。

ここではあえて、非形式的にこれを「証明」してみよう。これは厳密には「証明」というよりは、「説明」に過ぎないものかもしれないが、人類にとってより直感的で、説得力は充分であると私は期待している。

説明　外角の図をもう一度見ながら、多角形の辺に沿って歩くことを想像すると次のことがわかる。

外角とは、多角形の角の部分で方向転換する際の曲がる角度である。

先程の図の場合、反時計周りに図を一周すると、図に描かれた角度はまさに角での方向転換の角度である。逆回転すると、全ての角が、外角のもう一つの候補に変わる。

そう考えると、多角形の辺に沿って歩いて一回転すれば、最終的に曲がる角度の総和は 1 回転分になることがわかるはずである。よってその総和は 360 度になる。

系 1 多角形の内閣の和は

$$180(n-2)$$

になる。

証明　先程の証明を逆に辿れば良い。すなわち、一つ一つの内角と外角の和は 180 なのだから、内角の和 I と外角の和 O の和は、

$$I + O = 180 * n$$

であり、$O = 360$ なので、

$$I = 180 * n - 360 = 180(n-2)$$

になる。

何を持って数学の証明をエレガントと呼ぶかには、さまざまな考え方がありうるが、いくつかの考え方からこの「証明」をエレガントだと呼ぶことができると私は考える。

その考え方の一つは「その定理が自明であるような観点を提示することで、それを証明することができればそれはエレガントである」と説明することができる。

その考え方に沿えば、この「証明」を見れば、確かに「外角の和が 360 度であること」は自明であるように思えないだろうか？

もちろん、何が自明かは、その人の直感に依存する問題なので、これを押し付ける気はあまりない。一つの見方を提示しているだけだと思ってほしい。

ちなみに、ここらへんの話は深掘りすると、「証明とはなんだろうか？」という話に繋がる。

イアン・ハッキングは、デカルトの証明観とライプニッツの証明観の違いという興味深い話を『数学はなぜ哲学の問題になるのか』[2] で書いている。

それによると、デカルトにとって証明とは、「一目見て明らかなもの」であるという。これは、ここで言う「エレガントな証明」と概念的に近いことが見られるであろう。

それに対してライプニッツにとって証明とは「一つ一つのステップを確認することができるもの」と言うもので、さすが現代の記号論理学や計算数学の先駆けのようなものを考えていた人だけあるという感じで、こっちの方が現代の普通の証明の捉え方に見える。

実際、現代数学の証明は、それほど高度とは言えないものでも相当巨大なものになってしまい、デカルト的な意味での「証明」とはとても言えなくなってしまった。そこへ「こんなものは証明とはいない」と駄々を捏ねてもなんの意味もないだろう。

さらにライプニッツの証明観をより進ませるものとして、Coq などのような「証明支援系」の発達が挙げられる。これによって人間にとってとても読みづらく直感的でない証明でも、コンピュータによって正しさを保証することができる。その方法はまさにライプニッツが考えたように「一つ一つのステップを確認する」ということを、コンピュータの愚直さでどこまでも推し進めることによってである。

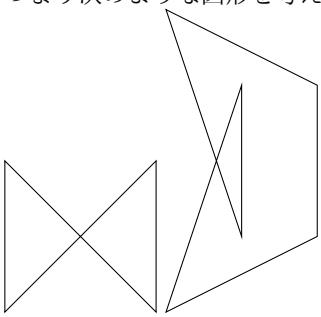
近い将来、数学者に「機械が確認しやすい証明」を書くスキルが求められることは充分に考えられると私は思う。これが人間にとって直感的にわかりやすい、デカルトのいう意味での「証明」、つまり「エレガントな証明」である可能性は非常に低いだろう。

ならば、エレガントな証明は無意味なのだろうか？　時代遅れの数学者の自己満足なのだろうか？

そうではないのではないか、というのがこの論考で言いたいことなのである。

そこで先程の外角の和についてもう少し深く掘り下げてみよう。この証明は明らかに凸多角形だけでなく、凹多角形についても成り立つが、辺の自己交差を認めたらどうなるであろうか？

つまり次のような図形を考えるのである。



これらについて先程の定理は明らかに成立しないが、左の図形の外角の和が 0 度であり、右の図形の外角の和が 720 度であることは、簡単にわかるであろう。辺に沿って進んだ際に何回転するかを数えればいいのだ。

もちろん、このような「自己交差を許す図形」について内角の和を論じることはできる。しかしこの場合、多角形において使った「三角形分割」の議論は素直には適用できない。外角への議論がそのまま適用できるのと大きな違いだ。

次に角の数を無限へと飛ばしてみよう。すなわち多角形ではなく曲線 c について考えてみるのだ[*9]。もちろん、角度という概念はなくなってしまうが、少し考えれば

- 曲線をパラメータ t で微分した関数は曲線の接線方向への速度
- 曲線をパラメータ t で 2 階微分した関数は速度の変化

となるので、この曲線の 2 階微分のうち接戦と直行する成分が「どれくらい方向転換して

[*9] 曲線の正確な定義については専門書にゆずるが、大雑把には連続関数

$$[0,1] \ni t \mapsto c(t) \in \mathbb{R}^2$$

と考えれば良い。これは曲線自体というよりは、曲線に沿っての移動と考えるとわかりやすいであろう。

いるか」、つまり曲線における外角の角度の対応物と考えられる[*10]。

そしてこの場合の総和は無限個の総和になるので、単なる和ではなく積分と考えるしかなくなる。

これは局所的な回転角の合計なので 2π の整数倍になる[*11]。なので、これを 2π で割った数値を**回転数**と呼ぶ。

これは大学数学の位相不変量の中でも特に早く習うものの一つであろう。

この概念によって、ある方向に回転する円をなめらかさを保ったまま逆回転する円に連続変形することはできないことが証明できる。そのような変形があったとした時、変形の各時点での回転数を計算すれば、0 から 1 へのずっと整数値の道が存在してしまうことになるが、そんなものはありえないからである。

このような概念を位相幾何学では**ホモトピー**という。

ちなみにこれを 3 次元の曲面に敷衍できるか、というのは興味深い問題である。すなわち

3 次元曲面に回転数のようなものを定義できるか。

という問題である。

これは、3 次元球面をなめらかなまま（自己交差は許して）変形して裏返ることができることから否定できる[*12]。すなわち、3 次元曲面に回転数のようなものが定義できたら、上記のような変形はありえないことになるからである。

さらにこの回転数を高次元に一般化しようとすれば、自然に**ホモロジー**の概念が生まれてくる。

このように、外角の概念は自然に基本的な微分幾何へと敷衍でき、それはホモトピーやホモロジーなどの位相幾何的な概念をいくつも生み出していく道筋の端緒になっているのだ。

このことから見ても、内角よりも外角の方が筋の良い概念であることがわかる。

そして、これらのことを知っているから、私は外角の和に関する証明の方がエレガントに見えるという側面があるのだろう。すなわち「さまざまな概念へと敷衍や一般化ができる概念なのでエレガント」という見方である。

私はここで、あえて逆も起こりうると主張したい。つまり「エレガントだからこそ、さまざまな概念へと敷衍や一般化ができる」という見方である。

最初の主張ではエレガントな証明とは「直感的に自明に思えるような証明」だった。

なぜ直感的に自明に思えるかというと、同義反復に聞こえるかもしれないが、直感が働くようにお膳立てされているからである。

先程の外角の和の証明の場合は、道に沿って一回転するという物理的な運動のイメージが援用されることによって、我々の直感が働くようにお膳立てされている。

エレガントな証明はそれだけ、我々のさまざまな経験への連想付けがされているわけである。

非論理的な仕方とはいえこのように様々な連想付けが可能なように整理されているということは、論理的にもさまざまな別の状況へと敷衍や一般化がされうることの経験的な証

[*10] なので、ここでは無限回微分可能な曲線、つまり**なめらかな曲線**について考えることにする。

[*11] このレベルまでくるとラジアンの自然さが強く出る。

[*12] Youtube に解説動画がある。`https://www.youtube.com/watch?v=BVVfs4zKrgk`

拠になりうるのではないか、と私は考える。この場合は、物理的な運動に連想づけられることによって、より滑らかな運動への一般化の道が開けている、と考えられるのではないか、ということだ。

数学においてある概念を定義する方法が複数存在することはよくある。例えば A という定義と B という定義が同値である場合、数学者は「A と定義してもいいし B と定義してもいい」と言うことがあるが、実は問題はそれほど簡単ではない。そのような状況でも数学者は「A という定義の方が良いか、B という定義の方が良いか」と悩みうる。

これはある意味これまで述べた美やエレガンスの問題と似ている。

どうしてそんなことを悩むのであろうか。

まず「定義 A と定義 B が同値」と言う場合、必ずなんらかの仮定が存在する。その仮定を外した時、ある概念の A と B はそれぞれ別の一般化となりうるのだ。

この時、果たして A がより良い一般化か、それとも B がより良い一般化か、と数学者は悩みうることになる（この場合の「より良い」はその数学者がどんな問題を考えているか、どんな一般化を求めているか、に依存する）。

その時に、数学者が頼りにするものの一つが、彼の直感である。

その直感の元になっているのは、それまで彼が数学においてさまざまな概念に触れた際の経験であり、それは多くの人の経験と大きくかけ離れているが故に、彼が直感的と思えるものは、多くの人にとってとても非直感的に見えるだろう（これは数学者でも、専門じゃない分野について触れた時には起こる）。

しかしそれでも、そこで起こっていることは、外角の和の照明に関して多くの人に対して起こったことと同じである。それまでの経験によって培われた直感と連想である。

「こちらの方が美しく思える」「こちらの方がエレガントに思える」と言う直感は、その概念がその数学者のそれまでの経験と豊かな連想で繋がっているが故に、より豊かな敷衍や一般化の可能性を持つことが期待される。

実際、そのようにして数学は発展してきたのではなかろうか、と私はここで主張したかったのである。

1.3　プログラミングにとって美とは何か？

さて、続いてプログラミングにおいての美やエレガンスについても語りたい。

まず結論から言うと、私にとってプログラミングにとっての美やエレガンスも、数学にとっての美やエレガンスとほぼ同様のものである。

つまり、

- 正しさが直感的に明らかである。
- 他の分野への敷衍や一般化の可能性を持っている

と言うことである。

プログラミングは数学よりも実用への指向が強いので、数学よりも美やエレガンスへの指向はずっと弱い。

それでも、プログラミングにとっても美やエレガンスは重要である、ということをこの文章で示したいと思っている。

さて、ここでも具体的な問題について語りたいが、業務で出てくるような問題で美やエ

レガンスについて語る準備が私の中で出来ていないので、少し不自然な問題設定になることについてはお詫びしたい。

この問題は『珠玉のプログラミング 本質を見抜いたアルゴリズムとデータ構造』[3] からとっている。

問題 1 長さ n の文字列を i 文字回転させるアルゴリズムを作れ。例えば、*"abcde"*という長さ 5 の文字列を 3 文字回転させると*"deabc"*と言う文字列になる。

もちろん、この問題を普通に解くのは簡単なので、少し制限をする。与えられた文字列以外で使っていい記憶領域は定数に限ることにする（つまり n や i が増えても、必要な記憶領域は増え続けてはいかない)。そして、実行時間は n に比例するようにする。その制限下でこの問題を解け。

例えば、i 文字分の記憶領域を確保して、文字列の前の部分をそこに保存し、文字列の後ろの部分を左に i 文字シフトさせて、文字列の後ろの部分に記憶領域に保存したデータを流し込んだとする。これによって回転自体は行えるし、実行時間も n に比例する。しかし使っている記憶領域は定数ではなく、i に比例して増えていってしまうので、問題の制限は満たしていない。

また 1 文字だけ回転させる関数を n 回適用することもできる。これで回転は行えるし、必要な記憶領域は 1 文字だけである。しかし、これでは実行時間は ni に比例するので、これも問題の制限は満たしていない。

プログラミングのできる方は、ぜひページを閉じて、一度考えてほしい。どうしてもなかなか手の込んだ方法が必要になるのがわかると思う。その手の込んだ方法の中に数学的な構造が見えてくることが、この問題の面白さである。

解答の一つは次のようなものである。

1 文字分の一時記憶領域 t を確保する。0 文字目を t に退避し、i 文字目を 0 文字目にコピーする。次に $2i$ 文字目を i 文字目にコピーする。これを何回か（k と置きます)）繰り返すと ki が n より大きくなるが、その時は ki を n で割った余り（つまり $ki \mod n$）の場所の値を $(k-1)i$ にコピーする。この値が $ki \mod n$ の値が 0 になった場合は、t に保存していた文字（つまり最初の 0 文字目の文字）を $(k-1)i$ 文字目にコピーする。

大学の学部程度の数学が分かる方、もっと具体的には剰余環 $\mathbb{Z}/n\mathbb{Z}$ が分かる方には n と i が互いに素なら、この時点で文字列が i 文字回転していることがわかるであろう。しかし全ての人が大学の学部程度の数学が分かるという仮定は成り立ちそうにないので、$n = 5$ で $i = 2$ の場合で説明すると。

- t　文字列
- □　abcde
- a　cbcde（0 文字目を t にコピーし、0 文字目に 2 文字目をコピー）
- a　cbede（2 文字目に 4 文字目をコピー）
- a　cbedb（4 文字目に 1 文字目をコピー）
- a　cdedb（1 文字目に 3 文字目をコピー）
- a　cdeab（3 文字目に 0 文字目ではなく t の文字をコピー）

となる。

これで n と i が互いに素でない場合（例えば $n = 6$ で $i = 2$ の場合）は、この 0 から

始まったループでは文字列の全ての部分を移動させることはできない。その場合は、1 から初めて同じループを辿る。このようにして 1 ずつスタート位置を変えていけば、必ず全ての場所を 1 回ずつ回転させることができる。これについてなぜか気になる方は、剰余環 $\mathbb{Z}/n\mathbb{Z}$ の数学について調べてほしい。それほど難しくはないが、高等数学のとても魅力的な入り口になっている。

$n = 6$ で $i = 2$ の場合で先ほどと同様に説明すると、

- t　文字列
- □ abcdef
- a　cbcdef（0 文字目を t にコピーし、0 文字目に 2 文字目をコピー）
- a　cbedef（2 文字目に 4 文字目をコピー）
- a　cbedaf（4 文字目に 0 文字目ではなく t の文字をコピー）
- b　cdedaf（1 文字目を t にコピーし、1 文字目に 3 文字目をコピー）
- b　cdefaf（3 文字目に 5 文字目をコピー）
- b　cdefab（5 文字目に 1 文字目ではなく t の文字をコピー。全ての文字を移動させたので終了）

このようにすれば、記憶領域は 1 文字分だけで、n に比例する実行時間で文字列を回転させることができる。

この解法は、剰余代数 $\mathbb{Z}/n\mathbb{Z}$ という美しい構造を使っていることが、そもそもエレガントであると私には思える。

ただ剰余代数 $\mathbb{Z}/n\mathbb{Z}$ を知らずに、この解決法を見出すのはちょっと難しく思えるかもしれない。そこで、もう少しプログラミングの定石に従った解放を探してみよう。

この場合「問題をより小さな問題に書き換える方法はないか」という観点で考えてみる。

n 文字の文字列の最初の i 文字を a、残りの $n-i$ 文字を b とし、文字列の連結を積としたモノイド[*13]を考えれば、元々の文字列は ab として表される。そしてこの問題は ab を ba に変えるアルゴリズムを見つけろ、というように言い直すことができる。

ここで簡単のために、$i \leq n-i$ と仮定する。$i > n-i$ の場合は回転方向を逆転させて、$n-i$ 文字反対方向に回転させると考えれば良い。そして b の右側の i 文字分の部分文字列を b_r、残りの左側の文字列を b_l とする。

このとき、同じ長さの a と b_r を交換するのには、一文字分の一時記憶領域で i に比例する時間で簡単に行える。すると、交換後の文字列は $b_r b_l a$ になっている。

もし $i = n-i$ ならば b_l は空文字列で $b_r = b$ なので、この時点で ba になり、i 字の回転が終了してしまう。そこで、$i < n-i$ であり、b_l は空文字列でないとしよう。

すると次に行うべきは、長さ i の b_l と長さ $n-i$ の b_r の連結した文字列 $b_l b_r$ を $b_r b_l$ にするという、同じ形だが、全体の長さは少なくなっている問題になっている。

このように再帰的に問題を簡単にすることを繰り返していけば、二つの部分の長さが等しくなって、アルゴリズムは終了し、その時点で回転が完了している[*14]。

そして全体の実行時間は、n に比例することも、一つ一つのプロセスの実行時間が二つ

[*13] 結合則の成り立つ積と単位元を持つ代数系。結合則とは $a(bc) = (ab)c$ であり、文字列の連結がこれを満たすのは自明。そして、この場合の単位元は空文字列である。ちなみにモノイドとしての文字列は**自由モノイド**と呼ばれるものになっている。

[*14] 全体の長さは狭義単調減少する自然数列であるので、無限降下列はありえない。

の文字列の短い方に比例することから分かる。

議論を追いかけやすくするために、$n = 5$ で $i = 2$ の場合で説明してみると

- abcde (2 文字左方向に回転させたい）
- decab（前 2 文字と後ろ 2 文字の交換）
- dec ab (今度は dec を 2 交換する文字左方向に回転させれば良いが、$2 > 3 - 2$ なので、右方向に 1 字回転させると考える）
- ced ab (dec の後ろ 1 文字と前 1 文字の交換）
- c ed ab （ed を 1 文字右方向に回転させる）
- c de ab（$1 = 2 - 1$ なのでここで終了）
- cdeab （元々の文字列を 2 文字左方向に回転させたものになっている。

さて、大学の学部レベルの数学を知っている方はこのアルゴリズムが**ユークリッドの互除法**そのものであることに気づいたかもしれない。そして、このアルゴリズムが剰余代数 $\mathbb{Z}/n\mathbb{Z}$ と関係が深いことにいろいろと思いを馳せたのではないかと私は想像している。

そういう意味で、このアルゴリズムも、

- 問題を再帰的に小分けにするというアルゴリズムの定石に従っていて
- しかも数学の世界への繋がりを感じさせる

という意味で、かなりエレガントに私には見える。

しかし、まだまだ複雑に見えるし、正しさも「直感的に明らか」と言うほどではない。

そこで、文字列のモノイドの代数系としての性質をもう少し調べてみよう。

「何かを代数系として考える」ということの良さは、複雑な操作を基本的な操作の組み合わせとして考えられる、ということである。「ブール関数は全て、NAND の組み合わせで構成できる」というのがその典型的例であり、これによってこのことによってスイッチング回路の設計は、職人技から科学に変わったのである。

それでは文字列のモノイドの積すなわち「文字列の連結」以外の操作で単純なものが何かないか考えてみる。文字列のモノイドは単純なものなので、それほど種類があることはないだろう、と想定しながら考えてみたとき、候補に上がるものの一つが「**文字列の逆転**」であろう。

$$\text{"abcde"} \mapsto \text{"edcba"}$$

文字列 a を逆転したものを、a の**転置**と呼び、**transpose** の頭文字をとって ${}^t a$ と書くことにする。

すると

$${}^{tt}a = a$$

が任意の文字列 a に対して成り立つことは簡単にわかる。

また、n 文字の文字列に対してこの操作を実現する一番素直なアルゴリズム（i 番目の文字と $n - i$ 番目の文字を交換していく）は、1 文字分の一時記憶領域のみを使い、文字列の長さに比例する実行時間しかかからないことも簡単にわかる。

次は転置と積の関係を見たい、と思うのが人情である。

$a = \text{"abc"}, b = \text{"de"}$ としたとき、$ab = \text{"abcde"}, {}^t a = \text{"cba"}, {}^t = \text{"ed"}, {}^t(ab) =$

”edcba” なので、次がわかる。

$$^t(ab) = {}^tb{}^ta$$

そして、これを任意の文字列 a, b に対して示すことができる。

すると、n 文字の文字列の左側 i 文字を a、残りの $n-i$ 文字を b としたとき、

$$^t({}^ta{}^tb) = ({}^{tt}b)({}^{tt}a) = ba$$

となり、i 文字回転が実現できる。

そして、この処理の過程で一時記憶領域は 1 文字分しか使っておらず、処理時間が文字列の長さに比例する処理を 3 回行っただけなので、全体としての処理時間も n に比例している。

そして、このアルゴリズムはここまでで紹介した他のアルゴリズムよりずっと簡単にプログラムすることができるし、ここまで読んだ人には正しさもかなり明らかになっていると期待できる。

これぞまさにエレガントな解答と呼ぶにふさわしい、と私には思える。

そして私がこのアルゴリズムに感じるエレガントさはこれでは終わらない。

l 行 m 列の行列と m 行 n 列の行列の積を考える。つまり、

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1m} \\ \vdots & & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{im} \\ \vdots & & \vdots & & \vdots \\ a_{l1} & \cdots & a_{lj} & \cdots & a_{lm} \end{pmatrix}, B = \begin{pmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ \vdots & & \vdots & & \vdots \\ b_{i1} & \cdots & b_{ij} & \cdots & b_{in} \\ \vdots & & \vdots & & \vdots \\ b_{m1} & \cdots & b_{mj} & \cdots & b_{mn} \end{pmatrix}$$

とした時の

$$AB$$

である。

ここで行列 A の**転置** ^tA を行と列の役割の交換、つまり、

$$^tA = \begin{pmatrix} a_{11} & \cdots & a_{i1} & \cdots & a_{l1} \\ \vdots & & \vdots & & \vdots \\ a_{1j} & \cdots & a_{ij} & \cdots & a_{lj} \\ \vdots & & \vdots & & \vdots \\ a_{1m} & \cdots & a_{im} & \cdots & a_{lm} \end{pmatrix}$$

とする。l 行 m 列行列 A の転置 ^tA は m 行 l 列行列になることは上を見ればすぐにわかるであろう。

そうしたとき、行列の積と転置の関係は

$$^t(AB) = {}^tB{}^tA$$

となる[*15]。

と言うわけで、先ほどと同じ恒等式が現れてしまった。

[*15] これは、線形代数の基本的な練習問題なので、線形代数を勉強中の人で証明したことがないならばぜひ挑戦してほしい。

ちなみに、文字列のモノイドと行列代数の両方でこの同じ式が現れる理由について、私は何らかの「深い説明」を全く持っていない。しかし、このように異なる場所で同じものに出会うことこそが、数学や通常のプログラミングの奥にある計算科学の（月並みな言葉で表現することを許してほしいのだが）ロマンであると私には思え、それは（数学の中の美やエレガンスについての節で説明したように）、追いかける価値のあるものだと思えるのだ。

業務に近いプログラミングの中で、このような方向性の抽象化はあまり推奨されない。

例えば **DDD（ドメイン駆動設計）**のような考え方は、「設計やプログラミングはできる限り顧客の言葉でなされるべきだ」と言う考え方である。DDD を強く押し出せば、「顧客の問題領域＝ドメインを数学や計算科学の都合で加工する」ことに対して抵抗を感じることになる。

DDD の有名な本『エリック・エヴァンスのドメイン駆動設計』[4] には、「閉じた操作」と言う説がある。これは原文では「Closure of Operations」であり、直訳したら「操作の閉包」とした方が良いように感じられる。しかし本文を読むと、エリック・エヴァンスが主張していることは、「ドメインの中で閉じている演算を探す」と言うことで、これはどちらかというと翻訳のニュアンスに近いものである。そして、まさにここにこそ、数学や計算科学と DDD が袂を大きく分ける場所になる。ドメインの中に演算を見つけたら、演算が閉じるようにドメインを拡張（操作の閉包をとる）してしまうことが、数学の定石なのだ。

だからと言って、顧客のドメインを忘れてしまうわけではなく、「顧客のドメインの問題を数学的に扱いやすいドメインに移して、そこで解決した後に顧客のドメインに解決策を引き戻す」と言う手順を辿って、最終的には顧客のドメインに戻ってくる。

これが可能であることを保証するのが、圏論でいう**普遍性**の力なのではないか、と私は考えている。

そして、プログラミングの世界でもこのようなやり方で作られた手法はたくさんある。

- 関係データベース（データベースに関する操作で閉じた代数系を関係代数として定義し、顧客のデータに関する問題を一度関係代数の問題に読み替える＝モデリングして、関係代数上でデータベースの問題を解決したのち、元々の問題に引き戻す）
- リアクティブ・プログラミング（イベント処理の問題を、イベントが流れていくパイプに対する join や merge などの操作の代数系の問題に還元する手法）
- 代数的データ型（リストやツリーなどのデータ構造を、データに対する特定な操作の閉包として定義する手法）
- 代数的エフェクト（副作用を足したり引いたりできる代数的な操作として考えようとする手法）

このように、プログラミングを大きく進展させる手法には、**普遍性**を目指した方向が生きているのに、現状（例え DDD を大きく打ち出していなくても）プログラミングの世界は、このような方向性に対して一定の抵抗を持っているように私には感じられる。

それはある意味ではとても正しいことで、なぜなら顧客からお金をもらって業務としてプログラミングを行う場合、最終目標は顧客の問題を解決することなのだから、「顧客のドメインが一番大事」と言う DDD の主張はとても正しいと言える。

しかし、「プログラミングの技術を覚えるよりも、顧客のドメインを理解することが大

切だし難しい（これは Twitter で実際に見た意見である）」のような意見を見ていると、バランス問題に思える。

そもそも顧客がプログラマに求めることが「顧客のドメインを理解すること」ではなく（これはどちらかと言うと当たり前で）、「プログラミングの知識」であることを見失っている気もするのだ。

多分、一つの問題は「ドメイン」と言う言葉で、顧客のドメインだけでなく、さまざまな顧客のドメインから普遍的なものを取り出した数学や計算科学の領域もまた「ドメイン」であるはずなのだ。顧客のドメインだけを「ドメイン」と呼ぶ今の言葉遣いではそこが見えにくくなってしまう。

『マルチパラダイムデザイン』[5] では、顧客のドメインを「**アプリケーションドメイン**」、開発側のドメインを「**ソリューションドメイン**」と読んでいるが、こちらの方がより良い言葉遣いのように私は思う。

これならアプリケーションドメインをソリューションドメインに加工し、その解決策をアプリケーションドメインに引き戻す、数学的な定石がよりよく表現できるからである。

参考文献

[1] 志村五郎. 数学の好きな人のために　——続・数学をいかに使うか. 筑摩書房. 2012

[2] ハッキング, イアン/ 金子洋之, 大西琢朗 訳. 数学はなぜ哲学の問題になるのか. 森北出版. 2017

[3] ベントリー, ジョン/ 小林 健一郎 訳. 珠玉のプログラミング 本質を見抜いたアルゴリズムとデータ構造. 丸善出版. 2014

[4] エヴァンス, エリック/ 今関 剛, 和智 右桂, 牧野 祐子 訳. エリック・エヴァンスのドメイン駆動設計. 翔泳社. 2011

[5] コプリン, ジェームズ O./平鍋 健児, 金澤 典子, 羽生田 栄一 訳. マルチパラダイムデザイン. 桐原書店. 2009

第 2 章

Tikz で双六を作ろう

淡中 圏

2.1　イントロダクション

Tikz とは LaTeX 上で図を描くためのライブラリーです。今回の The Dark Side of Forcing でも、図は全て Tikz で描かれています。

そして表紙の双六ももちろん Tikz で描かれています。

今回は Tikz で双六を描く簡単なチュートリアルをします。

2.2　なぜ Tikz で双六を描くのか

みなさんは、自由帳に双六を描いたことがあるでしょう。

楽しいですね。

しかし大人になって双六を描くことは非常に減ったのではないでしょうか？　私も減りました。

その理由はおそらく「自由帳に手描きで作った双六は修正や再利用がしにくい」と言うことではないかと愚考します。

そこでコンピュータの出番です。Word や PowerPoint などの MS Office 製品を使えば、修正や再利用は可能です。

しかし、Word や PowerPoint の弱点として、テキストデータではないのでバージョン管理がしにくい、と言う部分があります。皆さんもできる限りテキストエディタの外に出たくないというデリダ的欲望を持ち続けてることだと思います。

そこで Tikz です。

Tikz なら、全てテキストデータで双六を作ることができます。バージョン管理、修正、再利用、思いのままです[*1]。

皆さんもぜひ Tikz の使い方を覚えて、良き双六ライフをお過ごしください。

[*1] PDF にするのではなく、Web 上で双六を公開したいなら、**Mermaid** と呼ばれる Markdown 記法を使ってもいいかもしれません。Mermaid はグラフなどを表現するための記法で、フローチャートやシークエンス図、ガントチャートやクラス図、ER 図などをサポートしています。これほどさまざまなダイアグラムをサポートする割に双六のサポートがないのは不思議ですが、頑張れば Tikz と同様に双六を描くことができます。

2.3　Tikz の簡単な説明

ここではこの本で使ってるくらいの、Tikz の説明を書きます。

Tikz は、LaTeX ファイルのプリアンブルに

```
\usepackage{tikz}
```

と書くことで使うことができます。

また Tikz のさまざまな昨日はライブラリーとして分けて管理されています。それらはプリアンブルに

```
\usetikzlibrary{
  intersections,
  calc,
  arrows.meta,
  angles,
  quotes,
  graphs,
  positioning
}
```

のように、書くことで有効化できます。これはこの同人誌の LaTeX ファイルのプリアンブルに使われているものです。

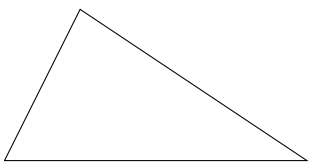
ここまでで準備は完了です。あとは、本文に

```
\begin{tikzpicture}
% 図形の記述
\end{tikzpicture}
```

上の「図形の記述」に Tikz のコマンドを打つことで、図形を描画することができます。

例えば、第 1 章でしたように、三角形を描画するためには、

```
\begin{tikzpicture}
\draw(0,0)--(1,2)--(4,0)--(0,0);
\end{tikzpicture}
```



とすればいいです。

Tikz のコマンドは必ずセミコロンで終わることに注意してください（私はよく忘れます）。

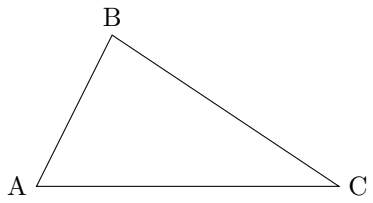
上のように閉じた図形を描くなら、

```
\begin{tikzpicture}
\draw(0,0)--(1,2)--(4,0)--cycle;
\end{tikzpicture}
```

と最後を cycle にしてもいいです。

また node コマンドによって、頂点のラベルを描画することもできます。

```
\draw(0,0) node[left]{A}
 --(1,2) node[above]{B}
 --(4,0) node[right]{C}
 --cycle;
```

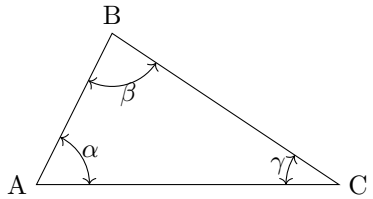


これはそれぞれの頂点にラベルをつけていますが、ラベルを方向を指定することによって、図を整形しています。

また、Tikz の angles ライブラリを使えば、角を表現することもできます。そのために、それぞれの頂点に内部的な名前をつけています（以下の a、b、c が内部的な名前です。）。

角のラベルや、半径、描画方法など、細かいオプションについては Web などを調べればわかります。

```
\draw(0,0) coordinate (a) node[left]{A}
 --(1,2) coordinate (b) node[above]{B}
 --(4,0) coordinate (c) node[right]{C}
 --cycle
 pic["$\beta$", draw=black, <->, angle eccentricity=1.2, angle radius=0.7cm]
   {angle=a--b--c}
 pic["$\alpha$", draw=black, <->, angle eccentricity=1.2, angle radius=0.7cm]
   {angle=c--a--b}
 pic["$\gamma$", draw=black, <->, angle eccentricity=1.2, angle radius=0.7cm]
   {angle=b--c--a};
```



また

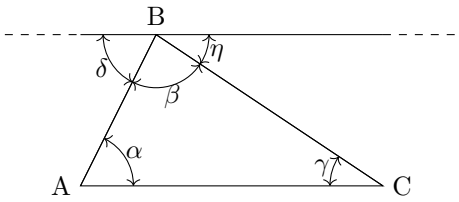
```
\draw[dashed]
```

と書くことで線ではなく点線を引くこともできるので、直線の一部などを表現できます[*2]。例えば、三角形の内閣の和の公式の証明に使われた図は

```
\draw(0,0) coordinate (A) node[left]{A}
  --(1,2) coordinate (B) node[above]{B}
  --(4,0) coordinate (C) node[right]{C}
  --cycle
  pic["$\beta$",
      draw=black,
      <->,
      angle eccentricity=1.2,
      angle radius=0.7cm]
    {angle=A--B--C}
  pic["$\alpha$",
      draw=black,
      <->,
      angle eccentricity=1.2,
      angle radius=0.7cm]
    {angle=C--A--B}
  pic["$\gamma$",
      draw=black,
      <->,
      angle eccentricity=1.2,
      angle radius=0.7cm]
    {angle=B--C--A};
\draw[dashed](-1, 2)--(0,2);
\draw(0, 2) coordinate (D)
  --(1,2) coordinate (B)
  --(0,0) coordinate (A)
  pic["$\delta$",
      draw=black,
      <->,
      angle eccentricity=1.2,
      angle radius=0.7cm]
    {angle=D--B--A};
\draw(4, 2) coordinate (E)
  --(1,2) coordinate (B)
  --(4,0) coordinate (C)
  pic["$\eta$",
```

[*2] もちろん他の線のスタイルもたくさんあります。

```
        draw=black ,
        <->,
        angle eccentricity=1.2,
        angle radius=0.7cm]
    {angle=C—B—E};
\draw[dashed](4, 2)--(5,2);
```

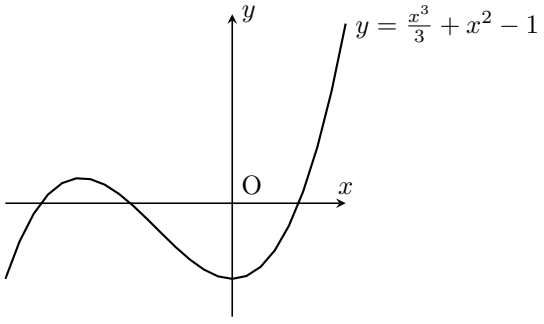


と書かれています。

また、数式を書いて、グラフを描画することもできます。例えば、

```
\begin{tikzpicture}
\draw[->,>=stealth ,semithick](-3,0)--(1.5,0)
  node[above]{$x$};%軸x
\draw[->,>=stealth ,semithick](0,-1.5)--(0,2.5)
  node[right]{$y$};%軸y
\draw(0,0)node[above right]{O原点};%
\draw[thick ,domain=-3:1.5]
  plot(\x,{(\x)^3/3 + (\x)^2 - 1})
  node[right]{$y=\frac{x^3}{3} + x^2 - 1$};
\end{tikzpicture}
```

と書けば、

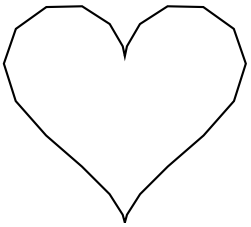


のようなグラフになります。

また、関数ではなくてもパラメトリック曲線を描くこともできます。せっかくなので愛を込めてハートを作ってみましょう。

```
\begin{tikzpicture}
\draw[thick ,domain=-3.14:3.14]
  plot(
      {1.6 * sin(\x r)^3},
```

```
        {1.3 * cos (\x r)
        - 0.5 * cos(2 * \x r)
        - 0.2 * cos(3 * \x r)
        - 0.1 * cos(4 * \x r)}
    );
\end{tikzpicture}
```

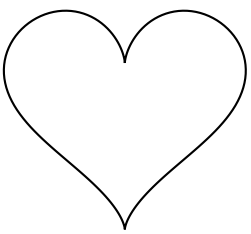


ちょっとこのハートはカクカクしてますね。多分、残業続きだったり、プロジェクトが炎上していたり、上司やマネージャーやクライアントから強いプレッシャーを受けたりして、イライラしているのでしょう。

そういう場合は、samples をしてあげることで、曲線の座標を計算する点を増やしてあげることができます。これを多めに指定すれば曲線が滑らかになるわけです。

やはりハートはトゲトゲしていない方がいいですね。

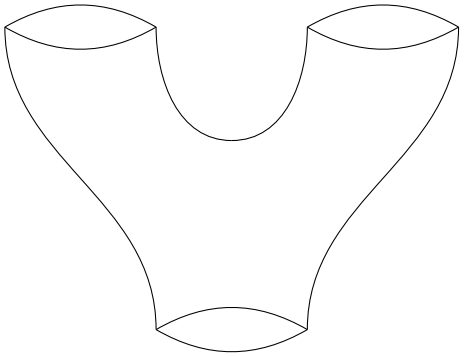
```
\begin{tikzpicture}
\draw[thick,
        domain=-3.14:3.14,
        samples=200]
    plot(
        {1.6 * sin(\x r)^3},
        {1.3 * cos (\x r)
        - 0.5 * cos(2 * \x r)
        - 0.2 * cos(3 * \x r)
        - 0.1 * cos(4 * \x r)}
    );
\end{tikzpicture}
```



関数のグラフは今回は使っていませんが、小説の図を書くためにパラメトリックな曲線の描画は使っています。

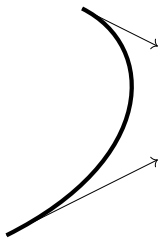
また、曲線については、視点と終点と指定して、それぞれの侵入角度を決めると簡単に書けます。これは小説でズボンを描くのに使いました。

```
\begin{tikzpicture}
\draw (-3,2) to [out=-30,in=-150] (-1,2);
\draw (-3,2) to [out=30,in=150] (-1,2);
\draw (1,2) to [out=-30,in=-150] (3,2);
\draw (1,2) to [out=30,in=150] (3,2);
\draw (-1,2) to [out=-90, in=180] (0,0.5);
\draw (1,2) to [out=-90, in=0] (0,0.5);
\draw (-3,2) to [out=-90, in=90] (-1, -2);
\draw (3,2) to [out=-90, in=90] (1, -2);
\draw (-1,-2) to [out=-30,in=-150] (1,-2);
\draw (-1,-2) to [out=30,in=150] (1,-2);
\end{tikzpicture}
```



もちろん、ドロウソフトなどでよく使われるベジェ曲線も使えます。

```
\draw [ultra thick] (8,1)
  .. controls (10,2) and (10,3.5) ..(9,4);始点，終点で方向点がそれ
ぞれ
    %(8,1)(9,4)とである超極太の次ベジェ曲線
    %(10,2)(10,3)"3"
\draw [->, very thin]     (8,1) -- (10,2);
\draw[<-, very thin] (10,3.5) -- (9,4);上のベジェ曲線の→つき方向線
を示した細線
    %""
```



これでこの本に出てくるくらいの Tikz の説明はおおむね終わりました。

最後に、双六を作るための説明をします。

```
\node[draw, circle, fill=white] (START) {START};
```

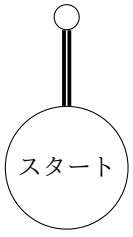
と書くことで、スタートの円を描き、それに START という名前をつけています。

もうそこにもう一つ円を描きます。テキストは空ですが、その後のために A と言う名前をつけます。。positioning ライブラリを使えば、START からの相対座標で位置を指定できます。

```
\node[draw, circle, fill=white]
  (START) スタート{};
\node[draw, circle, fill=white,
      above=1cm of START] (A) {};
```

そして、つけた名前を使って二つの円の間に線を引いていきます。double を指定することで、二重線にしています。

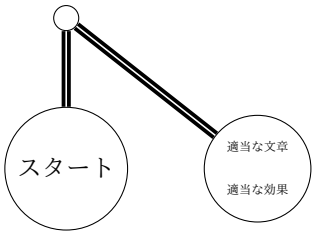
```
\node[draw, circle, fill=white]
  (START) スタート{};
\node[draw, circle, fill=white,
      above=1cm of START] (A) {};
\draw[double, line width=0.5mm]
  (START) -- (A);
```



文章を入れたいときは、字を小さくするための tiny コマンドをしています。また、改行させるときは、align を指定する必要があります。

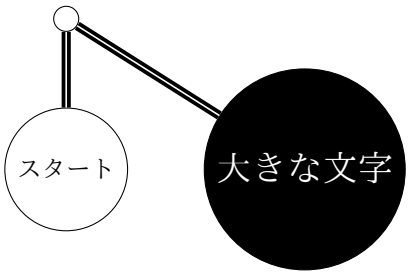
```
\node[draw, circle, fill=white](START)スタート
  {};
\node[draw, circle, fill=white,
      above=1cm of START] (A) {};
\draw[double, line width=0.5mm]
  (START) -- (A);
\node[draw, circle, fill=white,
```

```
      right=1cm of START, align=center]
(B) {\tiny 適当な文章 \\
     \tiny 適当な効果};
\draw[double, line width=0.5mm]
  (A) -- (B);
```



円を塗る色を変えたり、文章の色を変えたり、フォントを大きくしたりもできます。

```
\node[draw, circle, fill=white]
  (START) スタート{};
\node[draw, circle, fill=white,
      above=1cm of START] (A) {};
\draw[double, line width=0.5mm]
  (START) -- (A);
\node[draw, circle, fill=black, text=white,
      right=1cm of START, align=center]
(B) {\tiny 適当な文章 \\ \tiny 適当な効果};
\draw[double, line width=0.5mm]
  (A) -- (B);
```



これを繰り返していけば、双六ができますわけです。
それでは表紙のソースコードを示して、この章を終わりにします。

2.4 双六ソースコード

```
\begin{tikzpicture}
  \node[draw, circle, fill=white]
  at (current page.center)
  (START) スタート{};
  \node[draw, circle, fill=white, above=1cm of START]
```

```
(A) {};
\draw[double, line width=0.5mm] (START) -- (A);
\node[draw, circle, fill=white, left=1cm of START]
(B) {};
\draw[double, line width=0.5mm] (A) -- (B);
\node[draw, circle, fill=black,
      text=white, below=1.8cm of START]
(C) {\Huge Forcing};
\draw[double, line width=0.5mm] (B) -- (C);
\node[draw, circle, fill=white,
      right=2cm of START, align=center]
(D) {\tiny 学部入学。虚学先行。\\ \tiny 人生が加速。};
\draw[double, line width=0.5mm] (C) -- (D);
\node[draw, circle, fill=black,
      text=white, above=2cm of D]
(E) {\Huge the};
\draw[double, line width=0.5mm] (D) -- (E);
\node[draw, circle, fill=white, above=3cm of START]
(F) {};
\draw[double, line width=0.5mm] (E) -- (F);
\node[draw, circle, fill=black,
      text=white, above left=3.5cm of START]
(G) {\Huge of};
\draw[double, line width=0.5mm] (F) -- (G);
\node[draw, circle, fill=white,
      left=3cm of START, align=center]
(H) {\tiny 修士進学。 \\ \tiny 人生が加速。};
\draw[double, line width=0.5mm] (G) -- (H);
\node[draw, circle, fill=white,
      below left=4.5cm of START]
(I) {};
\draw[double, line width=0.5mm] (H) -- (I);
\node[draw, circle, fill=white,
      below=5.2cm of START, align=center]
(J) {\tiny 博士進学。 \\ \tiny 人生が加速。};
\draw[double, line width=0.5mm] (I) -- (J);
\node[draw, circle, fill=white,
      below right=5cm of START]
(K) {};
```

```
\draw[double, line width=0.5mm] (J) -- (K);
\node[draw, circle, fill=white,
      below right=2cm and 4.5cm of START]
(L) {};
\draw[double, line width=0.5mm] (K) -- (L);
\node[draw, circle, fill=white, right=5cm of START]
(M) {};
\draw[double, line width=0.5mm] (L) -- (M);
\node[draw, circle, fill=white, above=2cm of M]
(N) {};
\draw[double, line width=0.5mm] (M) -- (N);
\node[draw, circle, fill=white, above=2cm of N] (O) {};
\draw[double, line width=0.5mm] (N) -- (O);
\node[draw, circle, fill=white,
      above right=4.5cm and 3cm of START, align=center]
(P) {\tiny 満期取得退学 \\ \tiny 人生が加速。};
\draw[double, line width=0.5mm] (O) -- (P);
\node[draw, circle, fill=white,
      above=5cm of START, align=center]
(Q) {\tiny 妙な会社に潜り込む \\ \tiny 人生が加速。};
\draw[double, line width=0.5mm] (P) -- (Q);
\node[draw, circle, fill=white,
      above left=6.5cm of START]
(R) {};
\draw[double, line width=0.5mm] (Q) -- (R);
\node[draw, circle, fill=white, left=6cm of START]
(S) {};
\draw[double, line width=0.5mm] (R) -- (S);
\node[draw, circle, fill=white,
      below left=6cm of START, align=center]
(T) {\tiny 妙な会社から逃亡 \\ \tiny 人生が加速。};
\draw[double, line width=0.5mm] (S) -- (T);
\node[draw, circle, fill=black, text=white,
      below left=7.5cm and 3cm of START, align=center]
(U) {\Huge Vol.};
\draw[double, line width=0.5mm] (T) -- (U);
\node[draw, circle, fill=white,
      below=7.5cm of START, align=center]
(V) {\tiny 妙な会社から追放 \\ \tiny 人生が加速};
```

```
\draw[double, line width=0.5mm] (U) -- (V);
\node[draw, circle, fill=black, text=white,
      below right=7.5cm and 3cm of START]
(W) {\Huge 16};
\draw[double, line width=0.5mm] (V) -- (W);
\node[draw, circle, fill=white,
      below right=6cm and 6cm of START, align=center]
(X) {\tiny 妙な会社が潰れる \\ \tiny 人生が加速};
\draw[double, line width=0.5mm] (W) -- (X);
\node[draw, circle, fill=white,
      below right=2cm and 7cm of START, align=center]
(Y) {};
\draw[double, line width=0.5mm] (X) -- (Y);
\node[draw, circle, fill=white,
      above right=2cm and 7cm of START, align=center]
(Z) {\tiny 妙な会社を自分で作る \\ \tiny 人生が加速};
\draw[double, line width=0.5mm] (Y) -- (Z);
\node[draw, circle, fill=white,
      above right=6cm and 7cm of START, align=center]
(A2) {};
\draw[double, line width=0.5mm] (Z) -- (A2);
\node[draw, circle, fill=white,
      above right=8cm and 6cm of START, align=center]
(B2) {};
\draw[double, line width=0.5mm] (A2) -- (B2);
\node[draw, circle, fill=black,
      text=white, above right=9cm and 3cm of START,
      align=center]
(C2) {\Huge Side};
\draw[double, line width=0.5mm] (B2) -- (C2);
\node[draw, circle, fill=black,
      text=white, above=9cm of START, align=center]
(D2) {\Huge Dark};
\draw[double, line width=0.5mm] (C2) -- (D2);
\node[draw, circle, fill=black, text=white,
      above left=9cm and 4cm of START, align=center]
(E2) {\Huge The};
\draw[double, line width=0.5mm] (D2) -- (E2);
\node[draw, circle, fill=white,
```

```
        above left=7cm and 6cm of START, align=center]
  (F2) {\tiny 特に意味もなく加速};
  \draw[double, line width=0.5mm] (E2) -- (F2);
  \node[draw, circle, fill=white,
        above left=5cm and 6cm of START, align=center]
  (G2) ゴール{};
  \draw[double, line width=0.5mm] (F2) -- (G2);
\end{tikzpicture}
```

宣伝小説:ズボンを買いに行くためのズボンがないので慌てて用意する話

淡中 圏

この小説について

　この度、私こと淡中 圏の書いた短編の載る新しい SF 雑誌『新月』が刊行されることになりました。

　私は『冬の朝、出かける前に急いでセーターを着る話』という題名の小説を書きました。セーターの中が迷路のようになっていて、迷って大冒険する話です。

　他の方々も竹書房『ベスト SF2022』に選ばれるような人や、さまざまな賞を受賞したすごい人たちばかりで、作品もとても面白いです。

　買ってくれると非常に嬉しいです。

　せっかく良い時期に同人誌を出すので、ここで宣伝と行きたいですが、その短編をここに載せるわけには行きません。そこで代わりに似た感じの書き下ろし小説を載せておきます。もしこの小説を読んで、少しでも面白かったらぜひ買っていただけると嬉しいです。

　ちなみにこの小説のネタは Twitter 上での十三不塔さん（@hridayam）との会話から

思いついたものです。面白いネタを提供してくれたことを感謝します。十三不塔さんも上記の雑誌にとても面白い小説を書いていますので、ぜひ読んでください。

ズボンを買いに行くためのズボンがないので一時的に用意する話

出かけようと思ったがズボンがなくて困った。

「まだですか？」

そう急かされても困る。

そもそも今日は服を買いに行くのだ。服を買いに行くための服がないなんて、なんと使い古されたジョークだろうか。

ところで皆さんご存知のように、この世界は 10 次元の空間内で振動する微小なひもでできている。

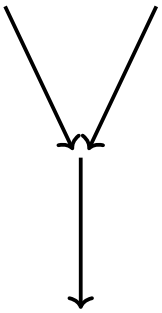
微小と言うのはこの場合**プランク長さ**のと言うことで、日常的な単位で無理矢理表せば 10^{-33}cm 程度だ。

そして、ひもには開いたひもと閉じたひもがある。

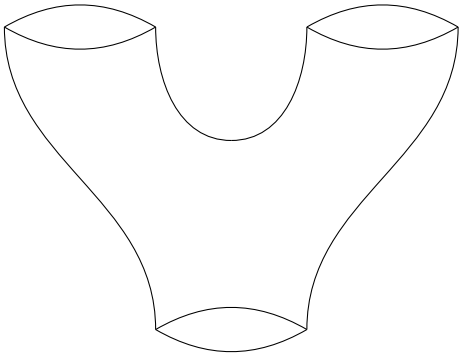
ブレーン宇宙理論によれば、世界の基本的な 4 つの力（電磁力、重力、弱い力、強い力）の重力以外を媒介するゲージ粒子は開いたひもだが、重力だけは閉じたひもである。そして、開いたひもの両端は全て**ブレーン**と呼ばれる 3 次元の膜に縛り付けられている。

重力を媒介する重力子だけが、この膜に縛り付けられておらず、それが重力だけが他の力よりも弱い理由というわけだ。

さて、ひも理論以前の量子力学で二つの粒子する際、次のようなイメージだった。



ひも理論では、これは次のように変わる。



お分かりであろうか。

ズボンである。

あとは穿くだけである。

しかし、口で言うほど、実際に行うのは簡単ではない。世の中の大抵のことと同じだ。

先程の図は上から下に向けて時間が流れていく図である。

つまり、たとえ重力子二つをぶつけて一つにしても、それだけでは穿くことはできず、ズボンを時間と空間の座標軸の中でぐるりと回転させなくてはいけないのだ。

これは普段から何度もやっていないと、なかなか難しい。

「電車、出てしまいますよ。次逃したら、次なかなか来ないんですから」

「ああ、分かってますってば」

思わず力が入り、ガタンと大きな音がする。

何かを崩したかと思い、思わず周りを見るが、何も崩れていない。何もかもが正常に見える。

いや、そうではない。異常なほど何も崩れていないのだ。エントロピーが増え続けるこの部屋では、必ず何かが崩れている。床に積み上げた本とか、箪笥にしまっていない服とか。

それが崩れていないと言うことは、答えは一つ。エントロピーの増大、すなわち時間の流れがストップしてしまっているのだ。

どうやら、間違えて、時間と空間の座標軸の方をぐるりと回転させてしまったらしい。それで時間の流れがストップしたのだ。

ということは、時間の流れの代わりに空間の流れができているはずで、現在我々の宇宙はどこかに向かって秒速 1 秒で流れているのだ。あまり長いことこのままにしておくのは得策ではないはずだ。

だが、今はせっかくなのでこの重力子のズボンを穿いてしまおう。見ればドアノブが回ろうとしている。どうやら様子を見るために入ってくるつもりだったらしい。危うくアンダーウェア一丁で時間と空間の座標軸を回す格好悪い姿を見られてしまうところだった。

と、重力子が一つになった側から足の指先を入れる。

「おや……これは？」

最近ダイエットをサボっていたせいか、どうやらこのズボンはスキニーすぎるようだ。腰まではとても入らない[*3]。このままでは重力子のひもをビリッと破いてしまう。その時

[*3] そもそもこのズボンは腰回りと裾の大きさがどちらも同じである。そんなズボンは不合理だと言わざるを得ないのではないか？

に何が起こるかは、最先端の現代物理学でも全く予測がつかない[*4]。

「うおおっと！」

無理矢理入れた両膝がプランク長さにまで圧縮され、バランスを崩す。思わず何かに捕まろうと手を伸ばす。

しかし、掴めるものは、時間と空間の座標系以外何もなかった。

そして、世界がまたグルリと回った。

「どうですか？　随分時間がかかりますね」

今それどころじゃ、と言おうとして気づいた。ここは自宅の部屋ではない。

狭い半畳ほどのスペースで、声はカーテンの向こうからしている。そして目の前の壁は鏡になっていて、そこに写った人物はズボンを手に持っていた。

なかなか小洒落たニッカポッカだ。

試着室？

混乱する頭で懸命に考えた結果導き出されたのは、どうやら先程時間と空間の座標軸を掴んだ際に、もう一度座標を回転させてしまったらしい。それで時間の流れが治ったのはいいことなのだが、回転だけでなく平行移動もさせてしまったのだろう、ついでに現在時刻が大きくずれたようなのだ。

そのせいで、部屋からここにくるまでの過程がすっ飛ばされてしまったというわけだ。

（それならそれでいいのだろうか？）

そう思いながら、手に取ったズボンを穿いてみる。サイズは合う。似合うかどうかは、自分のファッションセンスに自信がないのでどうとも言えない。

「サイズ大丈夫ですか？」

カーテンの向こうからまた声がしたので、カーテンを開く。

「ど、どうだろう？」

そう聞くと、

「いいんじゃないですか？　穿いてきたやつよりは？」

との答え。

穿いてきたやつ？　慌てて振り返って、壁の一つにかかったハンガーを見るが、何もかかっていない。

一体、何を穿いてここまで来たんだ？

そう頭の中で疑問が吹き荒れたが、答えを聞くのが怖くてそのままにしてしまった。

この日はそれ以外にも数着買ったが、帰りはそのズボンを穿いて帰った。

その後、何人かの知り合いに「あの時穿いてたのはすごかったね。また穿かないの？」と聞かれたが、一体どんなものを穿いていたのか確かめる勇気は、今のところまだない。

[*4] 一つ考えられることは、重力子が他のゲージ粒子と同じように開いたひもになってしまうと、他のゲージ粒子同様に両端がブレーンに囚われてしまうことだ。そうすると、重力だけが弱かった理由が消えてしまい、世界の重力が電磁力同様に強くなってしまうかもしれない。そうなったら宇宙の構造は大きく変わって、大混乱に陥ることが容易に予想できる。多分株価も大暴落するだろう。強い重力に引き寄せられて。

淡中 圏　会社がまた潰れました。

自分達で会社を作ることになりました。
あと、なんか商業作家としてデビューしてしまいました。

人生とは不思議なものですね。

8 月 27 日の SF 大会にはゲストとして参加させていただきます。メッセージカードを書け、と言われて「何を書けばいいんだ」と困惑したりしてました。福島で会える方はよろしくお願いします。

よく分からない自作 Web サイト　`https://tannakaken.xyz`

編集後記:

C100 だから出ておくか、という理由で無理やり作りました。次回は正直出るかわかりません。

ネタはあるので、もし書けたら出ようかな、くらいのつもりです。

技術書典に出るという手もありますね。

その時はまたよろしくお願いします。

【淡中　圏】

発行者 ： The dark side of Forcing
連絡先 ： http://forcing.nagoya
発行日 ： 2022 年 8 月 14 日

加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
Here Is The Math To The Dark Side
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ
加速せよ加速せよ加速せよ加速せよ