ペンローズタイル張りを描く基本的なアルゴリズムは以下のような感じです。まず、ペンローズタイル張りにとって基本的な定数である**黄金比**を

$$\varphi = \frac{1+\sqrt{5}}{2}$$

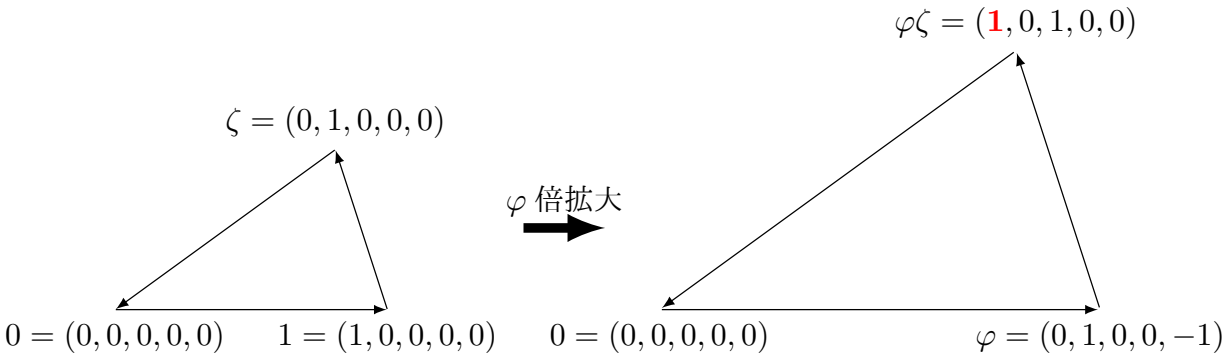
と定めます。黄金比が満たす基本的関係式

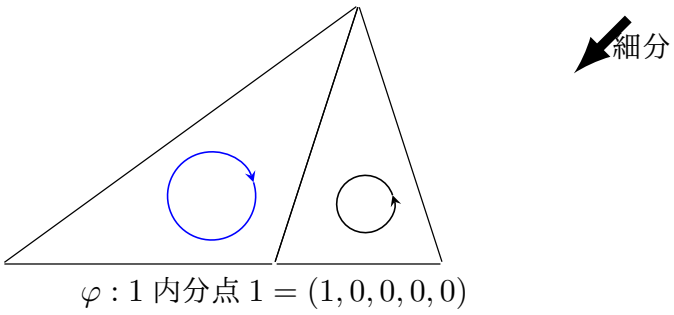
$$\varphi^2 = \varphi + 1$$

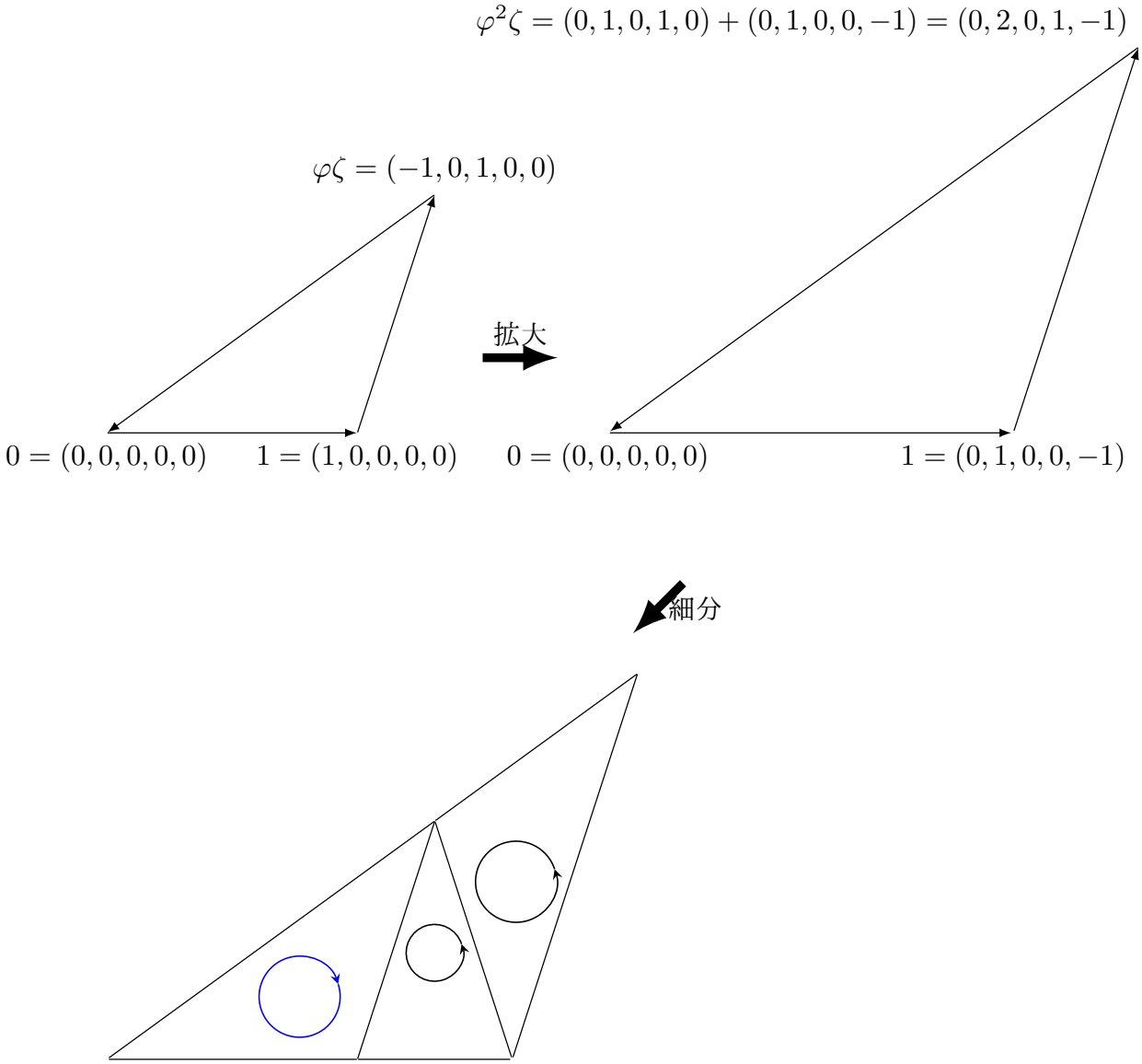
とその変形

$$\frac{1}{\varphi} = \varphi - 1$$

はこの後特に重要な働きをします。







平面 $\mathbb{R}^2$ を複素平面 $\mathbb{C}$ とみなすことにより拡大や細分、回転といった操作を複素数同士の掛け算や足し算によって表現できるので便利です。

$$\zeta = \cos\frac{\pi}{5} + i\sin\frac{\pi}{5}.$$

と定めます。このとき、関係式

$$\zeta^5 = -1, \quad \zeta^{-1} = -\zeta^4$$

と φ と ζ の間の関係式

$$\varphi = \zeta + \zeta^{-1}$$

が成り立ちます。よって

$$\frac{1}{\varphi} = \varphi - 1 = \zeta + \zeta^{-1} - 1$$

が成り立ちます。

ペンローズタイル張りに現れる各タイルの頂点の座標は整数 5 個からなるベクトル $\mathbf{x} = (x_0, x_1, x_2, x_3, x_4)$ を用いて

$$x_0 + x_1\zeta + x_2\zeta^2 + x_3\zeta^3 + x_4\zeta^4$$

と表現出来ます。浮動小数点ではなく整数で扱えることの良さはいろいろとあるかもしれませんが、特に基本的な操作である「拡大」とそれに続く「細分」がこの座標では著しく簡単に表現できる点が良いといえます。

拡大 複素平面上の点全体を φ 倍する操作です。これはもちろん複素数 x を φx に移す操作ですが、$\varphi = \zeta + \zeta^{-1}$ なので, $\mathbf{x} = (x_0, x_1, x_2, x_3, x_4)$ とするとき、5 次元座標を用いると、

$$\varphi x = (x_0, \ldots, x_4) \mapsto (-x_4, x_0, x_1, x_2, x_3) + (x_1, x_2, x_3, x_4, -x_0). \qquad (1)$$

と単純なシフト演算で表現できます。

細分 複素数 x と y が与えられたとき、それに対応する 5 次元座標を $\mathbf{x} = (x_0, x_1, \ldots, x_4)$, $\mathbf{y} = (y_0, y_1, \ldots, y_4)$ とします。線分 $x - y$ を $\varphi : 1$ に内分する点はもちろん

$$\frac{x + \varphi y}{\varphi + 1}$$

と表されます。

ここで拡大後の細分操作を考えます。つまり、x と y を φ 倍した点 φx と φy を $\varphi : 1$ に内分します。この操作で得られる点はもちろん、

$$\frac{(\varphi x) + \varphi(\varphi y)}{\varphi + 1}$$

であり、基本関係式 $\varphi^2 = \varphi + 1$ を用いると、

$$\frac{(\varphi x) + \varphi(\varphi y)}{\varphi^2} = \frac{1}{\varphi}x + y$$

と変形されます。$\frac{1}{\varphi} = \varphi - 1$ であったので、

$$\begin{aligned}\frac{1}{\varphi}x + y &= (\varphi - 1)\, x + y \\ &= \varphi x - x + y\end{aligned}$$

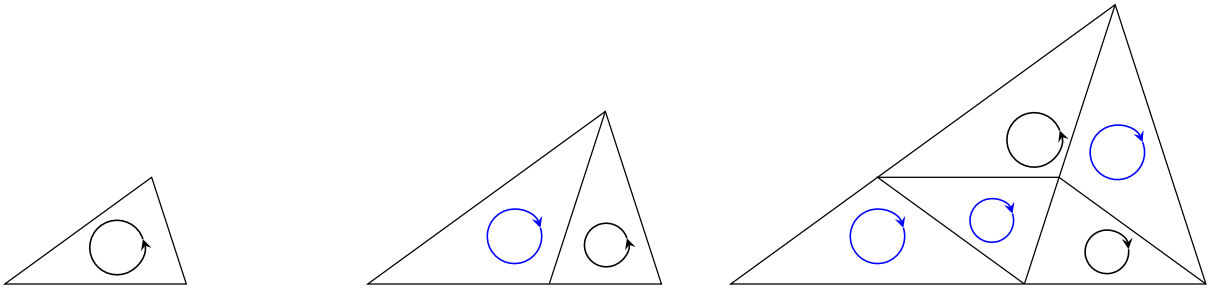
式 (1) を代入すると、この点を 5 次元整数ベクトルとして

$$(-x_4, x_0, x_1, x_2, x_3) + (x_1, x_2, x_3, x_4, -x_0) - \mathbf{x} + \mathbf{y}$$

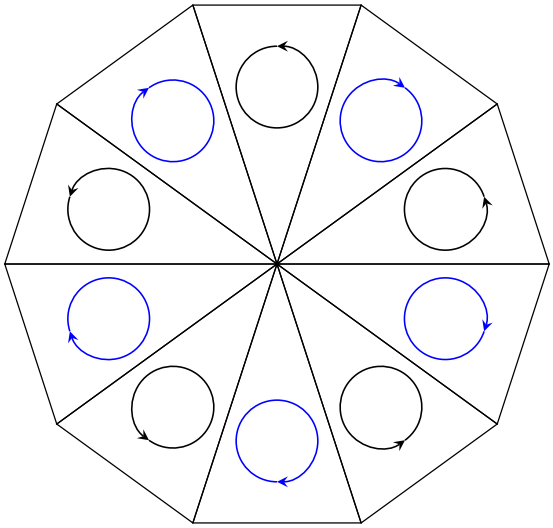
と比較的単純な式で表せます。

以上で拡大細分によって出来る新しい点を計算することが出来るようになりました。

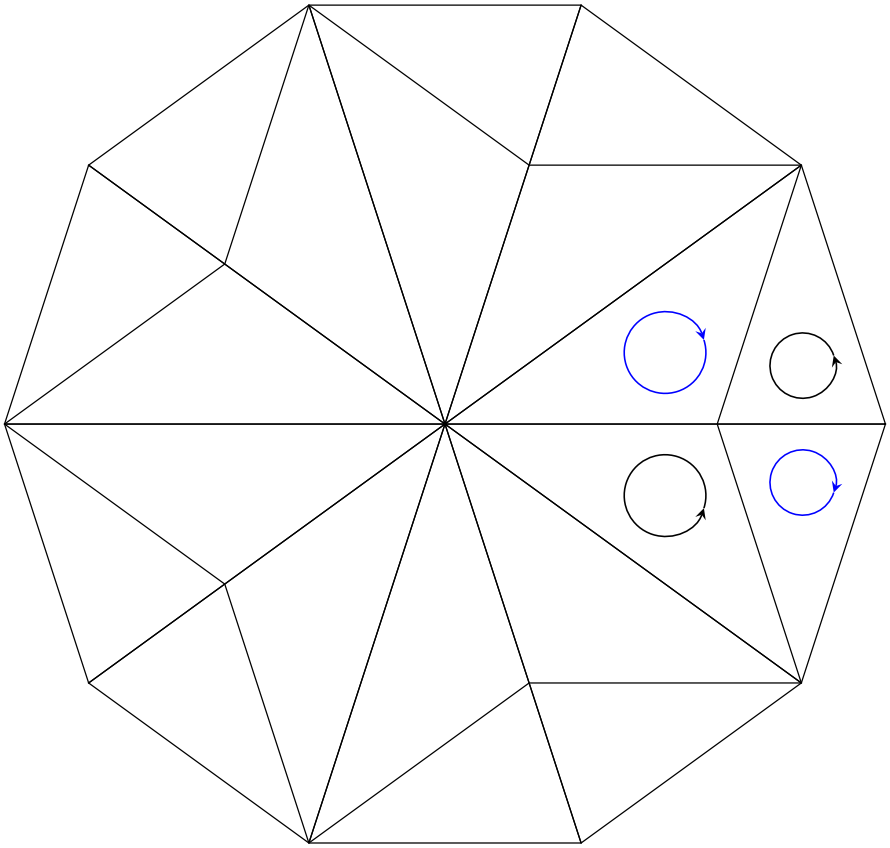
1 枚のタイプ A のタイルに対して拡大細分を 2 度施した様子を示します。



タイル張りを以下のようにして構成します。初期配置は下図のような 10 枚の A タイプのタイル 10 枚から構成されます。



ここで各タイルには向きがあり、1 枚おきに向きが反転していることに気をつけます。つまり、A 型のタイルを表向き、裏向き交互に配置していると考えます。これに拡大細分を 1 回適用すると、次のようなタイル張りが出来上がります。



1　プログラミング

1.1　言語

本当はこういうプログラミングには向いていないのかもしれませんが、完成品を手軽に使ってもらえるであろうから、javascript を使うことにしましょう。

1.2　データ構造

データ構造についてはどうしましょう？タイルは多角形です。多角形は辺と頂点からなります。頂点を 5 次元整数ベクトルで表すということまでは決定しています。

以下は仮の提案です。

頂点 v　長さ 5 の配列 (javascript では list と呼ぶ。)

頂点集合　頂点の配列 vertices、つまり配列の配列、2 次元配列です。

三角形　頂点集合を格納する配列 vertices のインデックスを 3 つ並べた配列で構成しま

す。このとき、配列の要素の並びに、つまり、頂点の並べ方に意味があります。これによって三角形が表向きか、裏向きかが決まるのですが、プログラミングを進める上ではあまり意識する必要はないと思います。

三角形の集合　A 型,B 型に分けて、三角形の集合として構成します。

ですから、最初状況は、下のような感じになります。

```
let vertices = [[0,0,0,0,0],
    [1,0,0,0,0],
    [0,1,0,0,0],
    [0,0,1,0,0],
    [0,0,0,1,0],
    [0,0,0,0,1],
    [-1,0,0,0,0],
    [0,-1,0,0,0],
    [0,0,-1,0,0],
    [0,0,0,-1,0],
    [0,0,0,0,-1]
         ];
let Atriangles = [[0,1,2],
     [0,3,2],
     [0,3,4],
     [0,5,4],
     [0,5,6],
     [0,7,6],
     [0,7,8],
     [0,9,8],
     [0,9,10],
     [0,1,10]
     ];
let Btriangles = [];
```

Listing 1　初期化

1.3　アルゴリズム

私達は既に頂点の φ 倍 (拡大) 操作を実装し、頂点 x, y を φ 倍した $\varphi x, \varphi y$ を $\varphi : 1$ に内分した頂点の計算方法を実装しました。

これらを用いて三角形の拡大細分操作を実装していきます。拡大操作は易しいのですが、拡大細分は少し戸惑うと思います。

A 型三角形では新しい頂点が一つ発生して、B 型三角形では 2 つ発生します。隣接する

異なる三角形で共通の新しい頂点が発生することがあり得ます。

最初は速度などはあまり気にしないで素朴に拡大細分を実行して新しい頂点が既に `vertices` に登録されているか確認して、なければ追加 `push` し、その index を新しい三角形の index として採用するというような操作で良いのかなと思います。