目 次

1 序論 **2**
1.1 タイリングとは 2
1.2 製作目的 2
1.3 試み 3

2 周期的タイリング **4**
2.1 エッシャー・タイリング 4
2.2 写像と等長写像 5
2.3 ２方向の並進 6
2.4 90 度回転と直交する並進 8

3 非周期的タイリング **9**
3.1 2 種のタイルからなるペンローズタイリング 10
3.1.1 カイトとダート 11
3.1.2 タイリング法 12
3.1.3 膨張・収縮 13
3.2 黄金比 14

4 ペンローズタイリングパズルの解説 **15**
4.1 パズルピースの仕組み 16
4.2 ボタン処理 18
4.2.1 パズルの型をかえるボタン 18
4.2.2 カイトとダートをそれぞれまとめて target に重ねるボタン . . 20
4.3 ピースの接触判定・吸着処理 22
4.3.1 選択されたピースを同じ形の target スペースに重ねる 26
4.3.2 隣に設置して配置できるピースにピタリとくっつく 27

5 まとめ **36**

1　序論

1.1　タイリングとは

平面を有限種類の図形で、重複も隙間もなく埋め尽くしたパターンをタイリング、あるいはタイル貼りと呼ぶ。そしてこのタイリングに使われる図形をタイルと呼ぶ。

図 1 に長方形の形をした一種類のタイルを使った例を 2 つ示した。

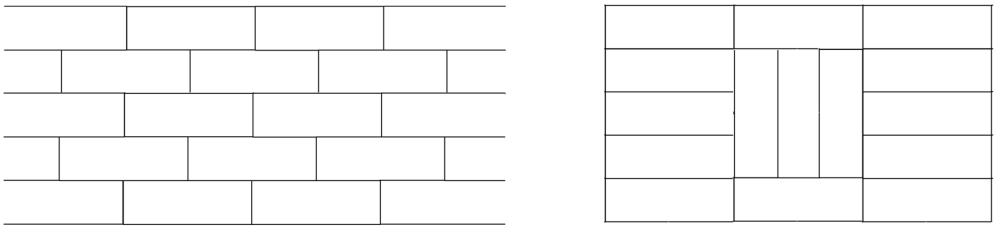

図 1: タイリング例　左図 (a) 右図 (b)

図 1(a) のタイリングはタイルが同じ向きに規則的に並べられて作られており、タイリング全体を平行移動によって動かして自分自身とぴったり重ねることができる。一方 (b) のタイリングでは 3 枚のタイルだけが縦の向きにおかれ、残りのタイルはすべて横向きに置かれい、平面全体にどのように平行移動を施しても、元のタイリングとぴったり重なることはない。

1.2　製作目的

本研究では、ペンローズ・タイリングの法則性、規則性を感覚的、視覚的に学べるパズル製作を行った。ペンローズタイリングは、一見どのような法則性でタイリングされているのかわからない複雑な形をしている。しかし実際には、ある法則性を満たしていれば、簡単に誰でもタイリングできるようになっている。この法則性の裏には複雑な計算や概念が隠れているが、このペンローズ・タイリングをパズル化することによって、複雑な計算や概念に触れることなく、感覚的にタイリングの法則性を理解できるのではないかと考えた。そのため、特にターゲットユーザーなどは絞らず、幅広い層の人に使ってもらえるようなパズル製作を行った。
パズルの遊び方は以下の二種類を用意した。

- 同じ形のピースを 1 箇所に重ねていくパズル
- 2 種のタイルを使ってのペンローズ・タイリングパズル

まず、同じ形のピースを 1 箇所に重ねていくパズルでは、ピースとなっているタイルの形を感覚的に理解してもらうところにある。また、初期画面には完成したパズルが表示されているため、完成されたパズルからピースをとりだし、どのようにタイリングされていたのかを視覚的に理解してもらうところにある。

2 種のタイルを使ってのパズルは、ピース同士がタイリングできる条件であればピタリとくっつくような処理が施されているため、視覚的にも感覚的にもタイリングの法則を理解することができる。

1.3 試み

本研究では、Adobe Flash8.0 を使用し、ペンローズ・タイリングの法則性、規則性を感覚的、視覚的に学べるパズル製作を行った。このソフトウェアでは Flash ムービーをつくることができる。このムービーの利点として、

- 低容量でかつ視覚効果のあるファイル作成が容易
- Web 公開が容易
- 一般的に普及している PC でほとんどのユーザーが使用可能
- 利用時の操作が容易

の以上 4 つが挙げられる。また。タイルを作成する際は、Adobe Illustrator CS5.5 を使用した。これはベクタ形式でデータ管理できるため画質が劣化しないことと、正確な図形をかくために適していたからである。

2　周期的タイリング

タイリングには二種類存在し、下記の通りである。

- 周期的タイリング (エッシャー・タイリング)
- 非周期的タイリング (ペンローズ・タイリング)

周期的タイリングとは、平行移動だけで作成可能なタイリング法であり、オランダの画家であるエッシャーが使用したタイリングである。一方非周期的なタイリングとは、平行移動しても作成が不可能であるタイリングを指す。これが今回の研究で題材にしているペンローズタイリングである。今回の研究をするにあたって、前知識として本章ではエッシャータイリングとペンローズタイリングの違いについて深めるため、この 2 つについて以下に記述していく。

2.1　エッシャー・タイリング

オランダの版画家エッシャーが得意とした作品の一つにタイリングアートがある。このタイリングアートでは鳥やとかげや人物などの複雑な形をしたタイルが画面に隙間なく覆われる。ときにはたった一種類の、ときには多種類の入り組んだ形のタイルがぴったりとかみ合って、隙間も重なりもなく平面が埋め尽くされる魔法のような描画方法である。

図 2: エッシャー・タイリング例 1

2.2　写像と等長写像

タイリングでは、タイルに相当する基本図形を場所と向きを変えながら平面へ敷き詰めて行く。これを言い換えると、1つのタイルから出発し、そのタイルのコピーを別の場所へ置く操作の繰り返しだと言える。このことからタイルのコピーを別の場所へ置くことは、もとのタイル内の点を別タイルの対応する点へ写像することとみなせる。

写像とは、二つの集合が与えられたときに、一方の集合の各元に対し、他方の集合のただひとつの元からなる集合を指定して結びつける対応のことである。函数、変換、作用素、射などが写像の同義語として用いられることもある。

タイリングに必要な写像は、タイルの形を変えない写像である。このような写像は等長という性質によって特徴づけることができる。それが等長写像である。

等長写像には以下の3つが例としてあげられる。

- 並進
- 鏡映
- 回転

エッシャータイリングはこの等長写像3つを考えれば作成可能である。しかしこの3つの等長写像を勝手な順序で施すだけでは、タイルが重なったり隙間ができてしまう。そのような不都合が生じないためにいくつかの制限を設け、それに従って写像していくことによりタイリング可能となる。

このようにエッシャータイリングにも規則性は存在し、基本の周期的タイリングとして計17の基本パターンが存在している。その中でも2方向の並進”と90度回転と直交する並進”の2種類を取り上げて説明する。

2.3　２方向の並進

正方形のタイルを上下左右に平行移動させて作るタイリングの仕組みとして、図2では一見ただ正方形のタイルが並んでいるように見えるが、このタイル１枚ずつに表と裏、上下左右の区別がついているとする。タイルが回転されたものなのか、反転されたものなのか、平行移動されたものなのかをわかりやすく区別するために、ここでタイル１枚ずつに”F”の文字をつける。これによりタイルが平行移動だけしてつくられているパターンだとわかる。(図3)

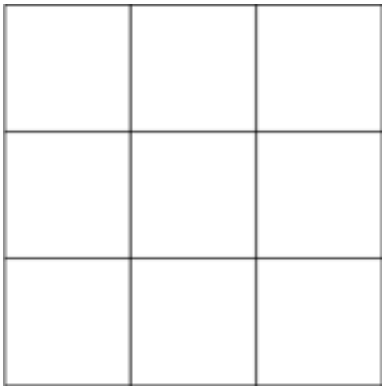

図 3: 正方形のタイルを基本として考えたタイリング

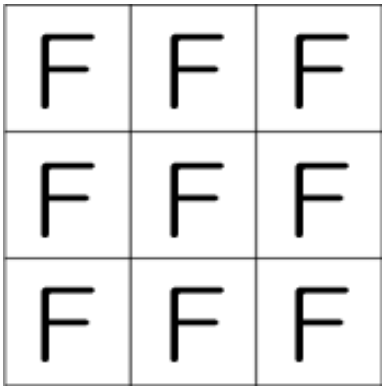

図 4: タイリング例１

次に、タイルがどう他のタイルと隣り合っているのか注目するために、タイルの”辺”について考えていく。
1 枚のタイルとその隣り合う 4 枚のタイルに着目する。中心のタイルの辺に、反時計まわりの向きをつけ、その順に abcd とラベルをつけていく。タイルを敷き詰める際、このラベルも一緒につけたまま移動させる。このときタイルそれぞれの辺の反対側にどのようなラベルが向きがくるかは、その周りの 4 枚のタイルから読みとることができる。図 4 からわかるように、a と c のラベルを持つ辺が逆向きに隣合い、b と d のラベルをもつ辺も逆向きに隣合う。この性質を理解することで、隙間もなく重なりもなく平面を覆い尽くすというタイリングの性質を保ったままタイルを変形することができる。
変形を施す時の注意として、1 つの辺を変形させた際、向かい合う辺も同じ変形をしなければならない。(a=c,d=b となっているため、a が変形すると c も変形する。db も同様である。)

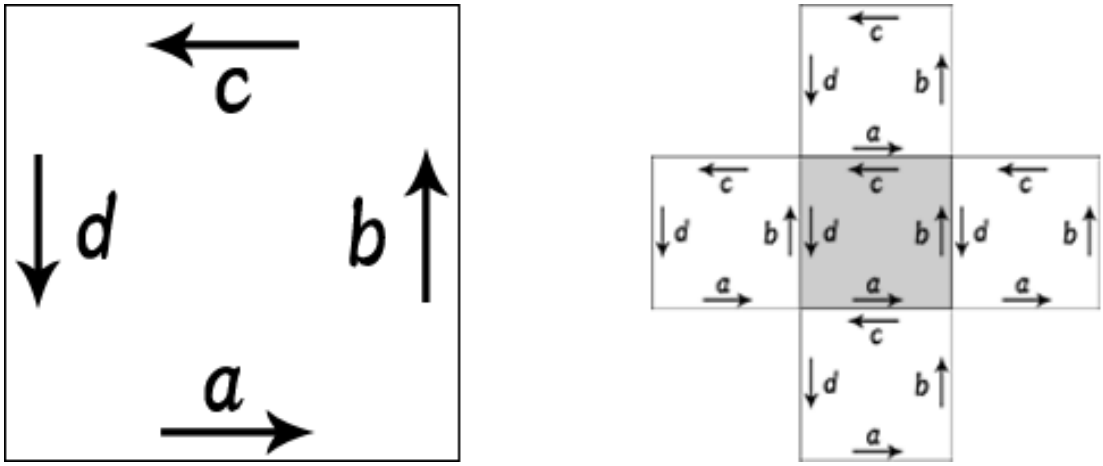

図 5: タイリング例 1 の辺の対応

2.4　90 度回転と直交する並進

この要領でまた別のタイリング法にも触れていく。中央のタイルに対して、左下の角を中心とする 90 °の回転を繰り返して、 4 枚のタイルを使って一周する面積が 4 倍の大きさのタイルが出来上がる。次にそれを縦横にその一辺の大きさを単位として平行移動させると、タイルのコピーが周りに置かれていく。この方法でも平面を覆い尽くすタイリングは可能である。

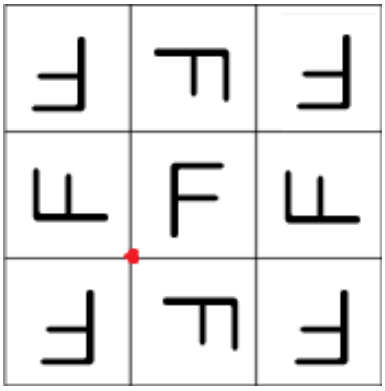

図 6: タイリング例 2

次にラベルに着目する。ラベルのうち方は 2 方向の並進と同じである。今回は a と d、b と c が向かい合っていることが分かる。 2 方向の並進と同じく、この規則にしたがって辺を変形していく。

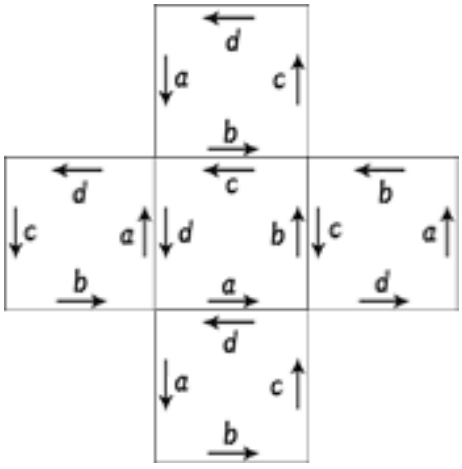

図 7: タイリング例 2 の辺の対応

このようなタイリングが周期的タイリングの一例であり、平行移動して作成できることができることがわかる。

3　非周期的タイリング

イギリスの物理学者であるロジャー・ペンローズが考案したものが、ペンローズタイリングである。このペンローズタイリングは上記で説明したエッシャータイリングとは異なり、単純なくり返し模様の出現を避ける一致規則でつくられており、隙間もなく重なりもなく、同じ模様を繰り返すこともない 1 組のタイルで平面を敷き詰めるタイリング法である。

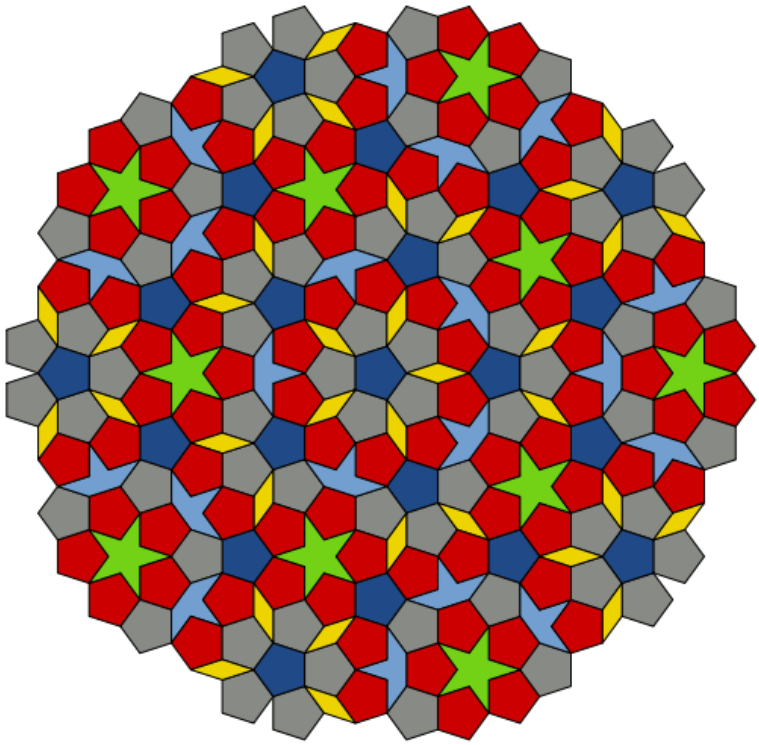

図 8: 4 種類のタイルからなるペンローズタイリング

ペンローズはまず最初に 6 種類のタイルからなる非周期的タイリングを 1973 年に発見して以来、この 6 種類を 4 種類 (上図)、2 種類と減らしたタイリングも発見した。今回は 2 種類のタイルからなるペンローズタイリングを中心に研究を進めたので、それを紹介する。

3.1　2種のタイルからなるペンローズタイリング

このタイルリングはカイト (凧) とダート (矢じり) と呼ばれる 2 種類の四角形ピースを使用してタイリングしていく。タイリング例として、中心点をカイトで埋めてタイリングするペンローズタイリング (無限の太陽の模様) と、中心点をダートで埋めてタイリングするペンローズタイリング (無限の星の模様) を下図に示す。

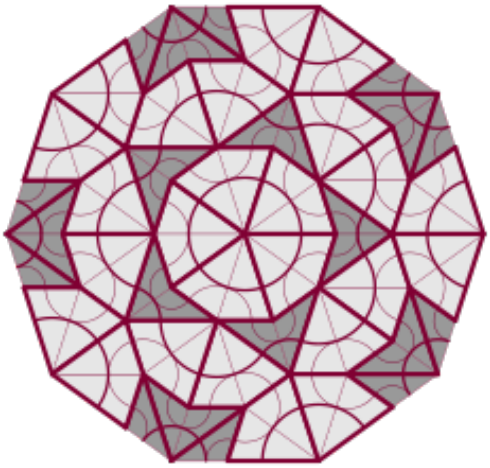

図 9: 無限の太陽の模様

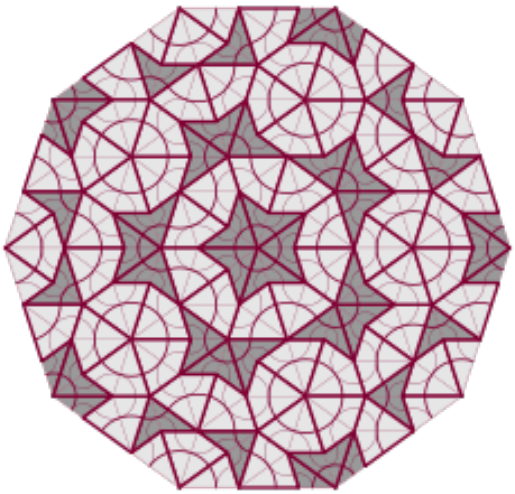

図 10: 無限の星の模様

3.1.1　カイトとダート

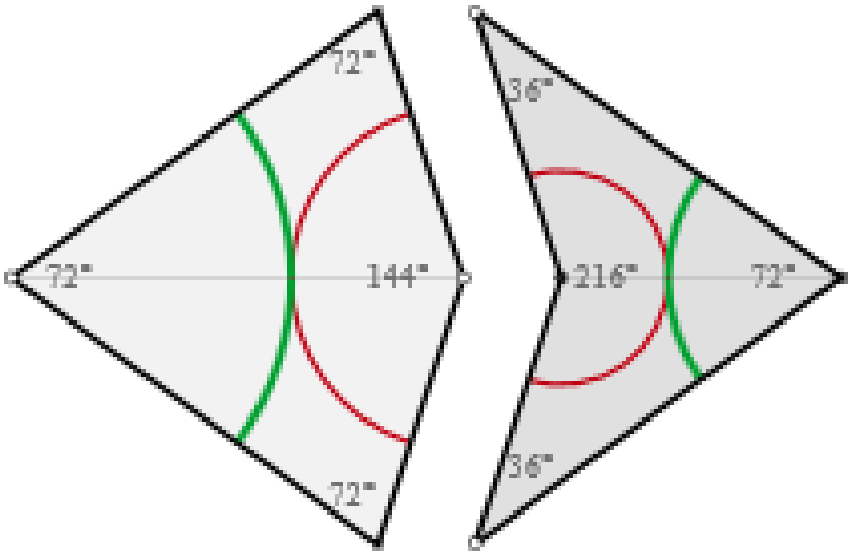

図 11: 左:カイト・右:ダート

カイトとダートの元となる形は 72 度、108 度の角からなるひし形である。このひし形の長い対角線を黄金分割 (ϕ (=1.61803398):1) し、鈍角の頂点と結ぶとカイトとダートを切り分けることができる。そして、カイトとダートの線分長は、それぞれ ϕ か 1 となっている。

また、上図での緑と赤の円弧はタイリングの補助線で、カイト・ダートの各辺と対称軸を黄金分割し、対称軸で円弧を合わせたときに同じ弧が重なるよう引かれたものである。正しくタイリングするとこの各円弧が途切れることなくつながり、頂点周りにタイリングしたときには閉じた円となる。逆に、円弧が途切れた場合、それは正しくタイリングできていないという証拠である。

このようにこのカイトとダートを作成するにあたって、深く関係してくることが黄金比である。上記以外でもタイリングで必要なカイトとダートの枚数比や、カイトとダートの面積比もまた黄金比となっている。この黄金比については後に詳しく記述する。

3.1.2　タイリング法

上記の通り、カイトとダートはひし形からできた図形だということがわかった。しかし、タイリングをする際にこの 2 つをくっつけて元のひし形にするタイリング方法はこのタイリングでは禁止されている。ペンローズタイリングは単純なくり返し模様の出現を避ける一致規則でつくられているため、周期的タイリングが可能なひし形のタイリングは避けなければいけない。

この他にも特別に規則を置かれているタイリング法がある。それは、ダートの凹部分には、必ずカイトが 2 枚くっつくというものである。この形はエースと呼ばれるが、3 枚のタイルが 1 つのカイトを形作っているため、愚者の凧とも呼ばれている。

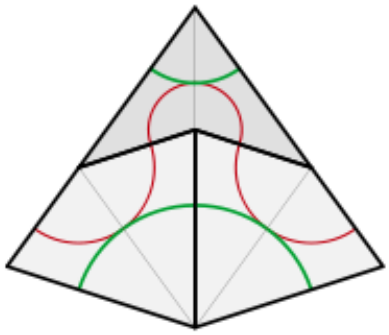

図 12: エース (愚者の凧)

この 2 つの規則を守ってタイリングしていくと、頂点周りをうめるタイリング法はエースも含めて計 7 通りであることがわかる。

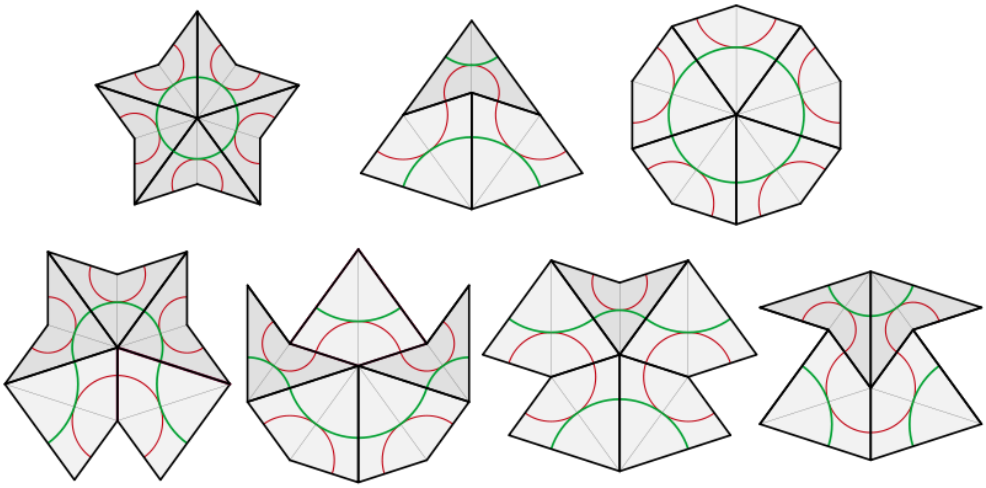

図 13: 頂点周りをうめるタイリング法計 7 通り

3.1.3　膨張・収縮

ペンローズタイリングには膨張と収縮という、同じ面積内で各タイルを拡大、縮小しても、タイル同士が重なることなく隙間なく埋め尽くされていく性質を持っている。これを各タイルを拡大させていくことを膨張。各タイルを縮小させることを収縮と呼ぶ。この頂点周りをカイトのみで、ダートのみでうめる太陽と星のタイリングは、膨張・収縮する際に面白い関係を持っている。
それは収縮・膨張を繰り返すたび、2 つのタイリングが交互に現れることである。

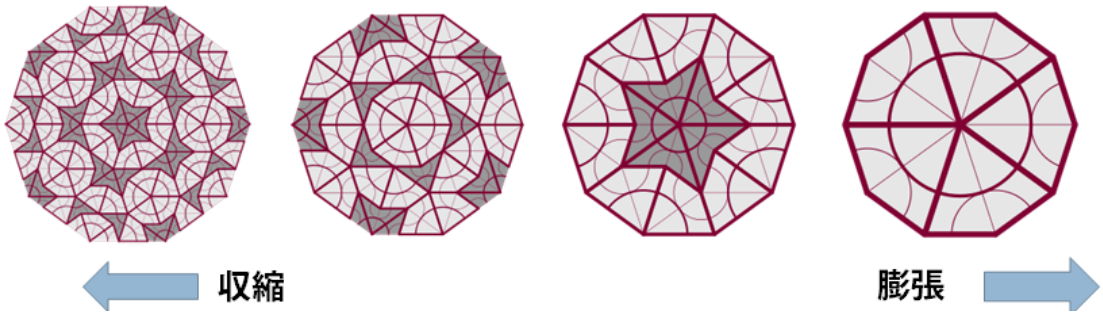

図 14: 膨張・収縮

3.2　黄金比

黄金比とは古くからギリシアの数学者ユークリッド (紀元前 300 年頃) の「ユーグリッド原論」の中にもみることができる、最も有名な比率である。比率の式は以下のようになっている。

$$1 : \frac{1+\sqrt{5}}{2} \tag{1}$$

ローマ時代にはウィトルウィウスが「建築十所」(紀元前 25 年前) で人体の比率に注目して建築理論を構成し、黄金比に該当する比率を取り上げた。その後レオナルド・ダ・ヴィンチが挿絵を描いた L. バチオーリの「神聖比例論」(1509 年) では、黄金比が「神聖比例」と呼ばれている。

実際にこの黄金比はギリシアにあるパルテノン神殿や、4500 年以上も前に作られたピラミッドなど、歴史的建造物に使われている。また、現代ではたばこの箱や新書のサイズにまで黄金比は幅広く使われている。

4　ペンローズタイリングパズルの解説

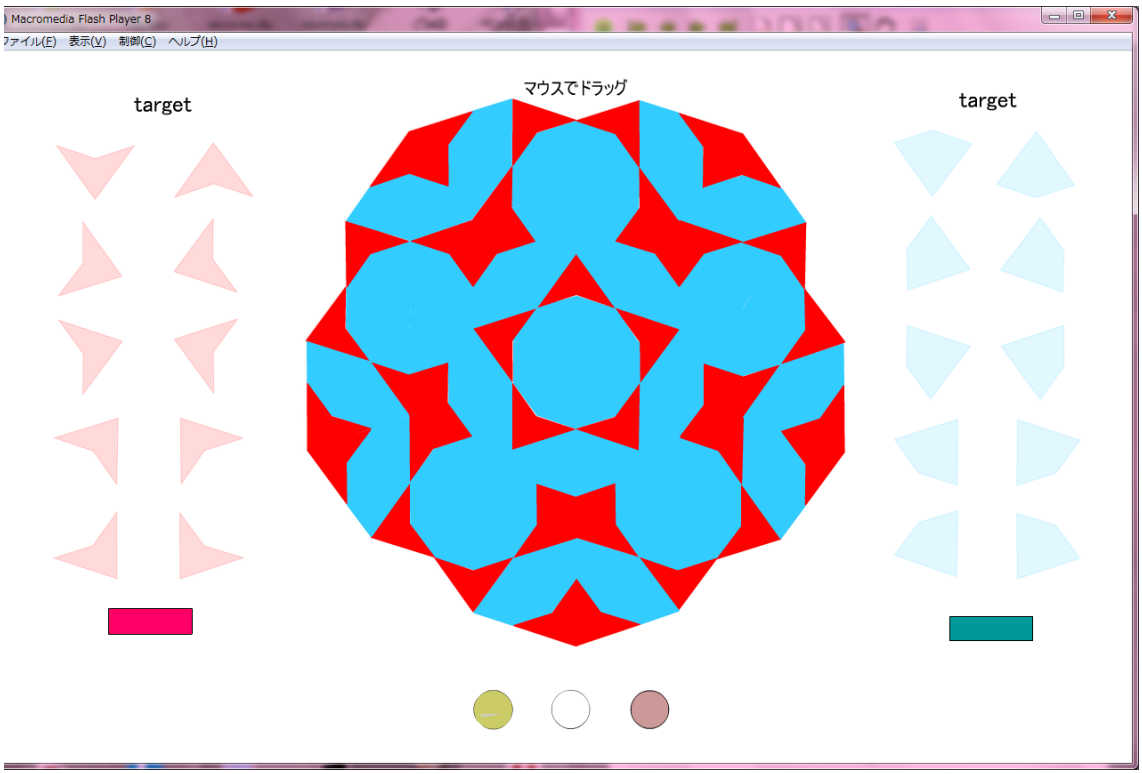

図 15: パズルゲーム画面

2 種類のタイルからなるペンローズタイルをパズルとしてマウス操作で自由に動かすことができる。それによりユーザーが視覚的・感覚的にペンローズタイリングの仕組みを学べるパズルである。選択したパズルのピースと同じ形の target ピースに重ねることや、型の上でペンローズタイルをパズルしていくことができる。なお、用意した型は無限の太陽の模様と無限の星の模様の計 2 種類である。また、自由にタイリングできるよう、ボタン操作で型を表示しないでパズルをすることも可能である。

4.1　パズルピースの仕組み

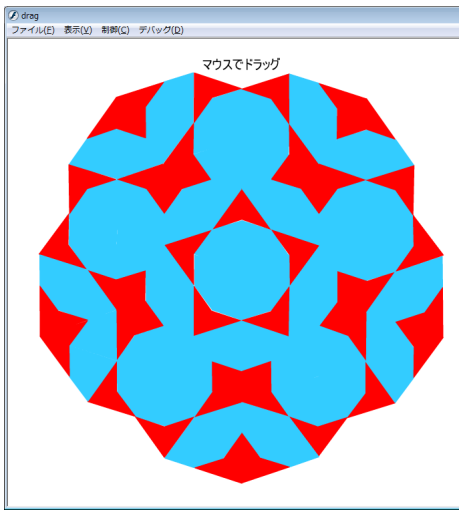

図 16: ペンローズタイル初期状態

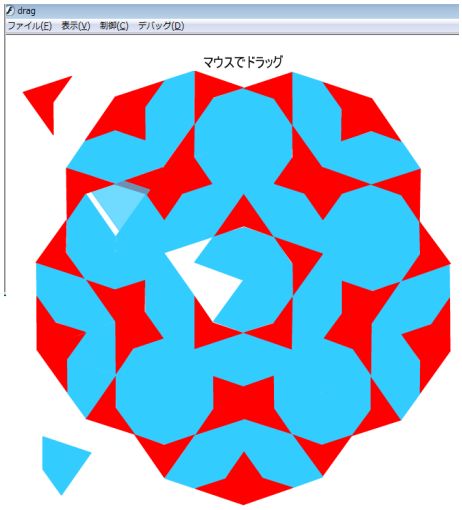

図 17: ペンローズタイルタイル移動後

パズル開始画面では、カイトとダートを最初から 1 枚ずつくっ付けた状態 (無限の太陽の模様) で表示させている。これはマウスのドラッグ・ドラッグ・リリースに対応しており、クリックしたタイルが自由に動かせるようになっている。クリック・ドラッグ中はタイルの透明度を下げてどのタイルを選択しているかわかりやすくした。

なお、カイトとダートは Flash の図形作成ツール (図■■) で作成したものであり、プログラムで作成した図形ではない。

図 18: 使用ツール (線ツール・バケツツール)

作成する際は、データ挿入した画像をトレースするかたちでタイルを作成した。トレースする際使用した画像は Adobe Illustrater で作成したものである。

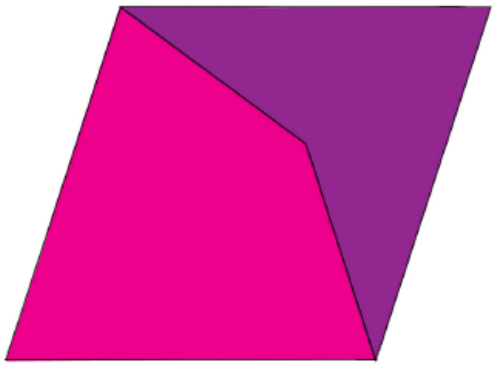

図 19: Adobe Illustrater で作成したカイト (ピンク) とダート (紫)

使用するタイルが揃ってからは、タイルをそれぞれムービークリップ化し、ActionScript を使用してパズル作成のプログラムを行った。

4.2　ボタン処理

パズル画面には、以下 2 種のボタンを設置した。

- パズルの型をかえるボタン
- カイトとダートをそれぞれまとめて target に重ねるボタン

ボタンオブジェクトはピースの作成方法と同じく、Flash 内のツールを使用して作成した。ボタン命令はすべて ActionScript で定義づけされている。

4.2.1　パズルの型をかえるボタン

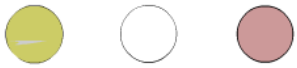

図 20: 型変更ボタン (緑:太陽・白:型無し・ピンク:星)

パズルの下にはパズルをしやすいように元になる型が表示されている。画面下にある 3 つのボタンはそれぞれ、無限の太陽の模様 (左緑ボタン)、無限の星の模様 (右ピンクボタン)、型なし (真ん中白ボタン) が設定されている。ボタンはマウスクリックで型を切り替えることが可能である。初期の型は、初期画面のパズルが無限の太陽の模様となっているため、同じくパズルの下には無限の太陽の模様の型が設定されている。

ボタンがクリックされると画面外座標に置かれている型オブジェクトが、画面中心部に移動するよう座標処理が行われる。型を表示しないボタンが選択された場合は、型オブジェクトが画面外座標に移動するよう処理されている。

図 21: 無限の太陽の模様の型 (初期設定)

図 22: 無限の星の模様の型

4.2.2　カイトとダートをそれぞれまとめて target に重ねるボタン

図 23: target にまとめて重ねるボタン (赤:ダート・青:カイト)

散らばったタイルをまとめて target に重ねるようなボタンを 2 種類用意した。ダートタイルをまとめて target に重ねる赤いボタンと、カイトタイルをまとめて target に重ねる青ボタンボタンを target スペースの下に設置した。

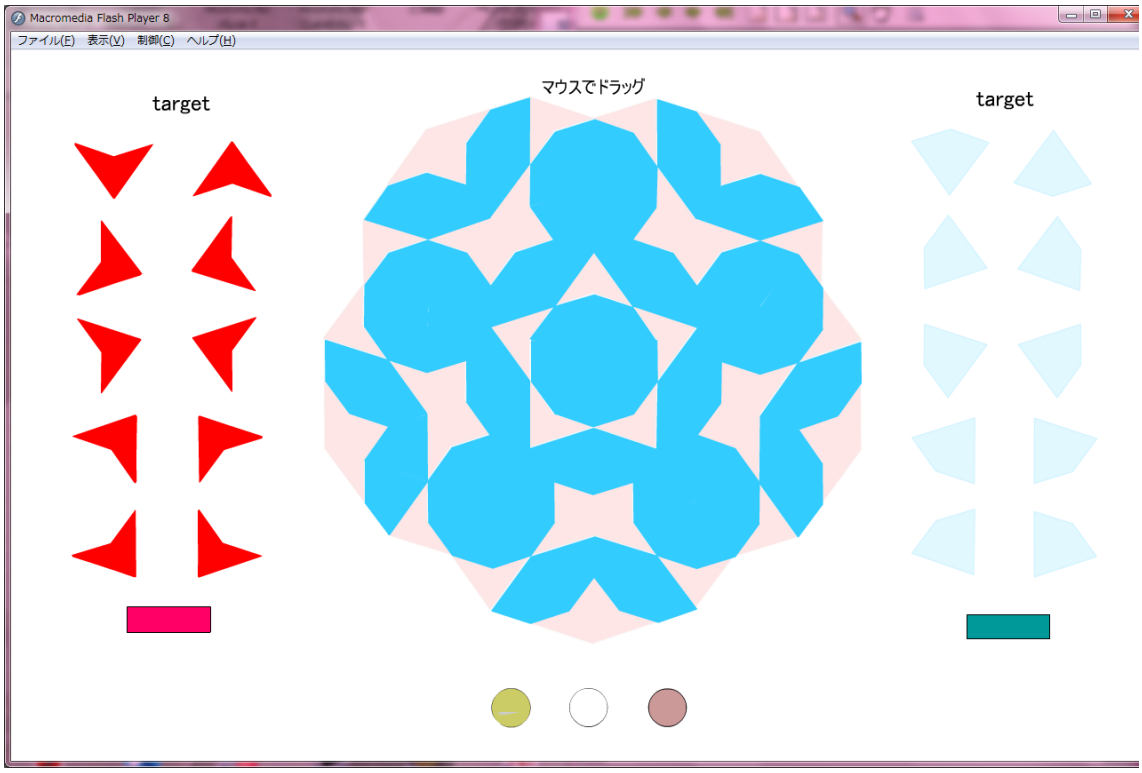

図 24: 赤ボタンが押された時のパズル画面

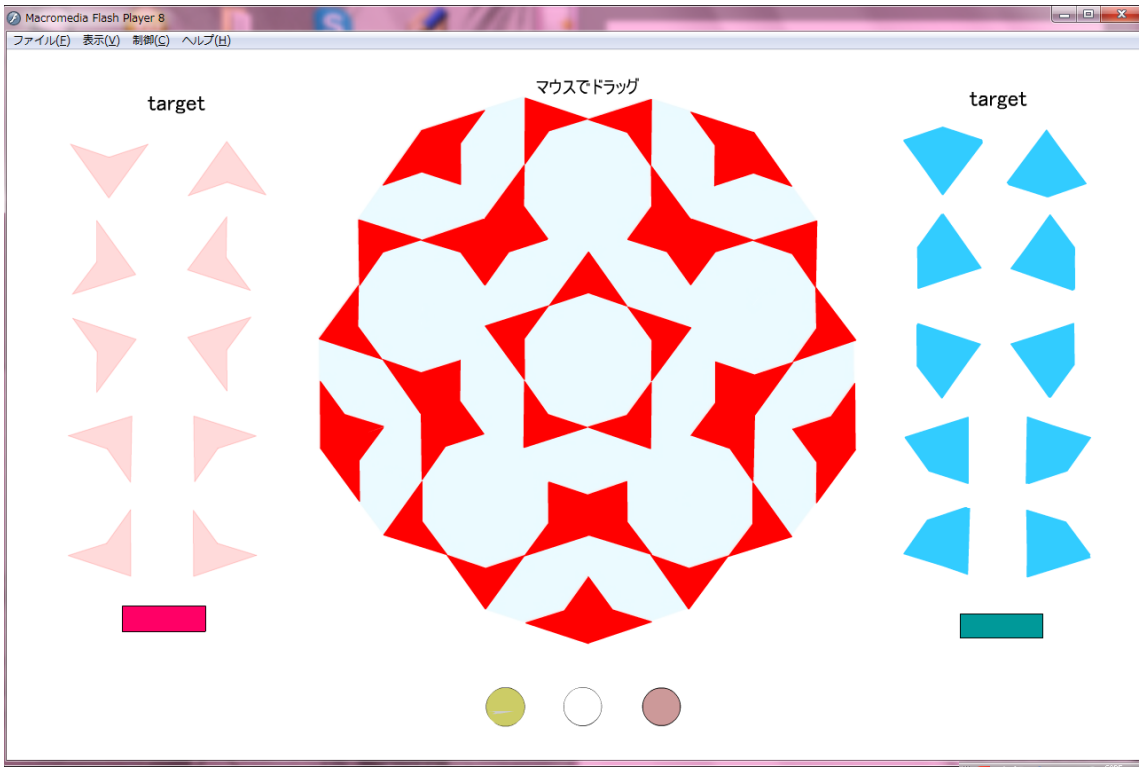

図 25: 青ボタンが押された時のパズル画面

ボタンがクリックされると、ピースオブジェクトの座標がそれぞれが対応する target オブジェクト座標へ移動する吸着処理が行われる。それぞれのピースと target 同士の吸着処理については次の小節で詳しく説明していく。

4.3　ピースの接触判定・吸着処理

パズルゲームには、以下の 2 つの接触判定・吸着処理を施した。

- 選択されたピースを同じ形の target スペースに重ねる
- 隣に設置して配置できるピースにピタリとくっつく

接触判定には、ActionScript の MovieClip.hitTest() 文を使用して接触判定を行う。接触判定はオブジェクトの形にそって行われるのではなく、そのオブジェクトの境界ボックス (図■■) で接触判定が行われる。

吸着処理は、クリック選択したオブジェクトの座標を、対象オブジェクト座標と対応させ座標移動させる。このときの処理で使われるオブジェクト座標は、境界ボックスの中心座標が使われている。

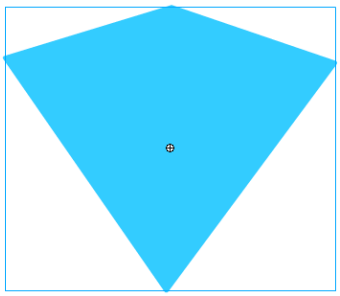

図 26: 境界ボックス (青色外枠)

ピースの接触判定・吸着処理のプログラムで重要になってくるものが、オブジェクト 1 つ 1 つに定義されるインスタンス名である。このインスタンス名は、名前に重複がある場合うまくプログラム処理が行われないため、オブジェクト 1 つ 1 つが被らないよう名前を定義する必要がある。なお、インスタンス名定義はプログラム上ではなく、オブジェクトプロパティ上で行った。

図 27: オブジェクトプロパティ

今回はtargetとなるオブジェクト20種類と、パズルのピースになるオブジェクトすべてにインスタンス名定義した。インスタンス名は以下の手順で名前をつけた。

- カイトピースには「k」、ダートピースには「d」のローマ字をつける
- targetオブジェクトはkとdの後ろにピース番号(*1)を、「_」の後ろにはtargetを付ける。
- ピースオブジェクトにはkとdの前にピース番号を、「_」の後ろにピースの個別番号(*2)を付ける。

*1：ピース番号は、カイトとダートそれぞれ同じ形のピースに共通してつけられる数字(下図参照)。
*2：個別番号は、同じ形のピース内でそれぞれを識別できるようつける数字。

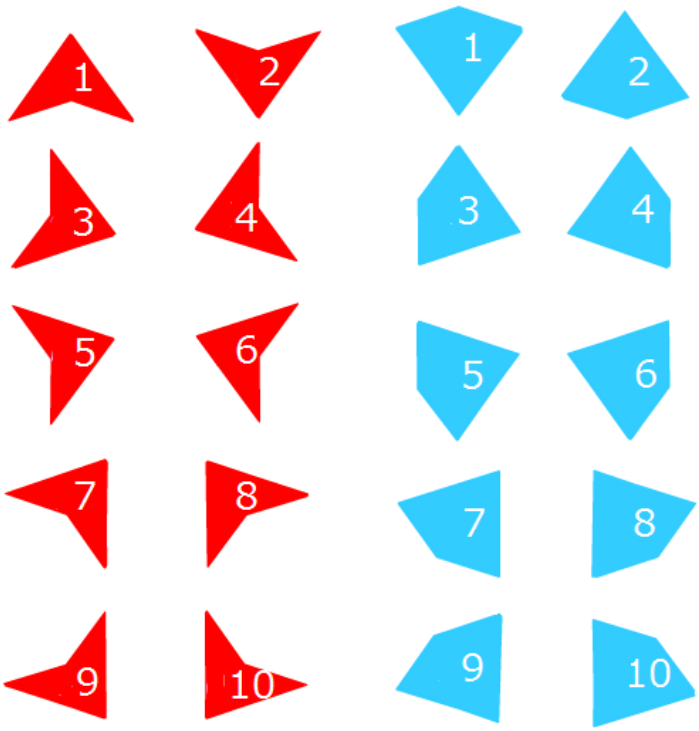

図 28: ピース番号

以上に従って定義した各オブジェクトのインスタンス名は下図の通りである。

図 29: target オブジェクトのインスタンス名

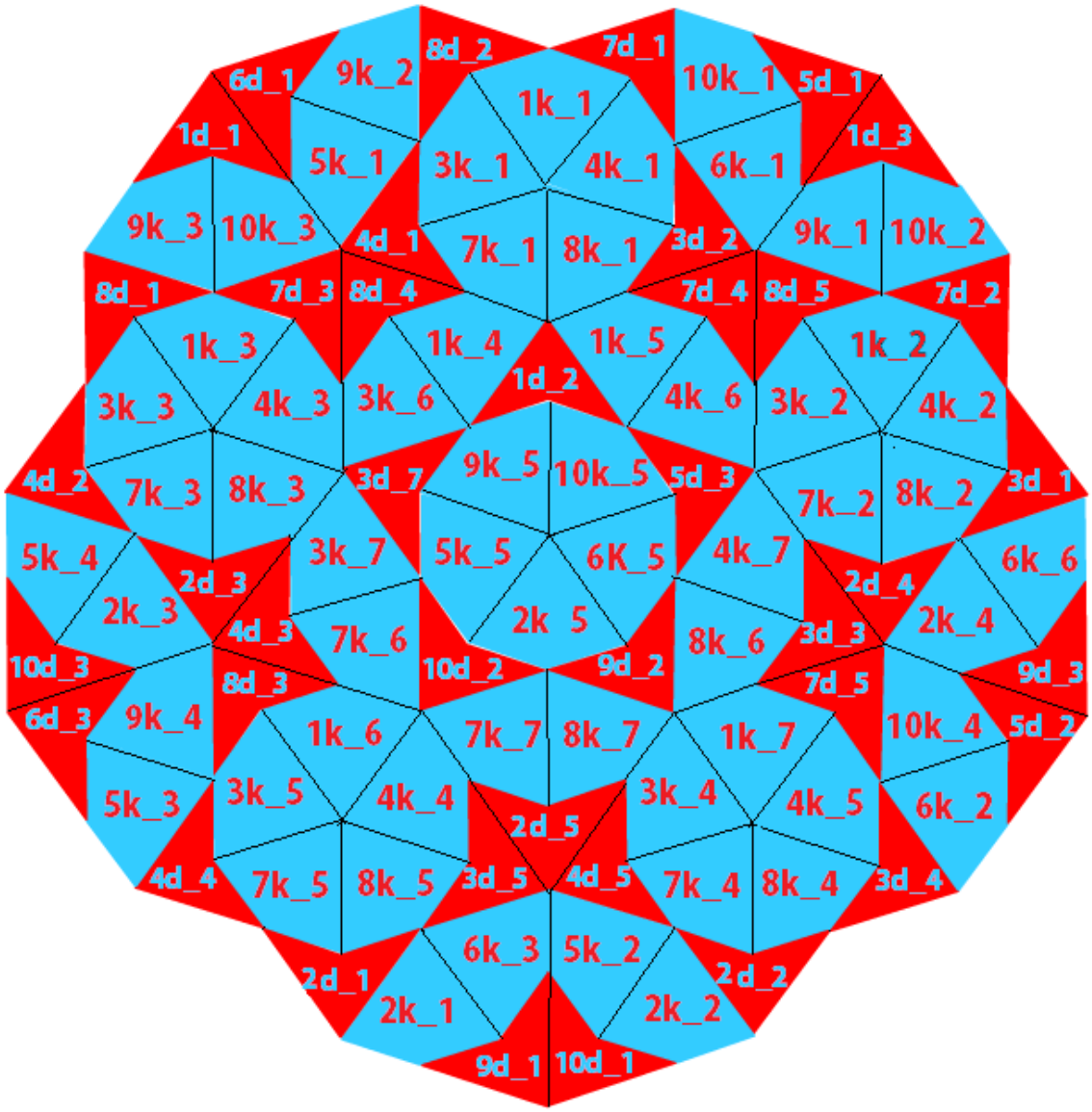

図 30: パズルピースオブジェクトのインスタンス名 (初期配置パズル)

4.3.1　選択されたピースを同じ形の target スペースに重ねる

図■■が選択されたピースを同じ形の target スペースに重ねた画面である。クリック選択しているピースオブジェクトの境界ボックスと、対応する target オブジェクトの境界ボックスが接触する場合のみ、吸着処理が施される。

吸着処理は、i をピース番号、j を個別番号として、インスタンス名「ik(d)_j」(i=1〜10,j=1〜7) のクリック選択されたピースオブジェクトが、インスタンス名「k(d)i_target」の target オブジェクトと接触したとき、クリック選択されたピースオブジェクトが target オブジェクトと同じ座標になるよう処理が施されている。

そして前小節で挙げた、カイトとダートをまとめてそれぞれの target に重ねるボタンは、ボタンが押されるとこの原理で吸着処理が行われる。

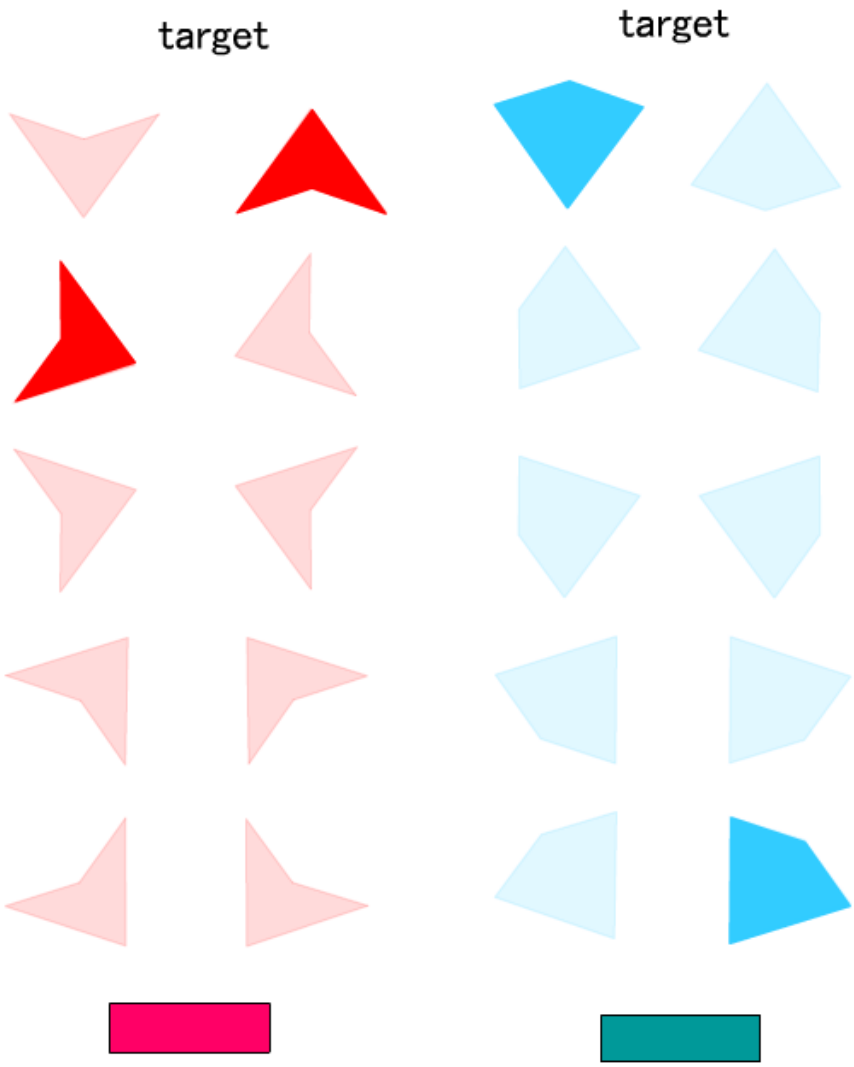

図 31: 選択されたピースを同じ形の target スペースに重ねる

4.3.2　隣に設置して配置できるピースにピタリとくっつく

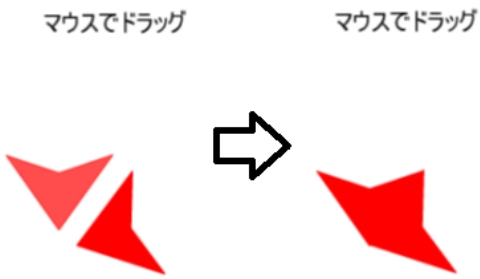

図 32: 隣に設置して配置できるピースにピタリとくっつく

上図がクリック選択したピースが、相手ピースにピタリとくっついたときの例である。これはクリック選択されたピースオブジェクトと、接触するピースオブジェクトに定義された各辺の角度データ配列から値を取り出し、タイリング可能かの判定を行う。

判定が true の場合、判定された辺を生成する 2 つの頂点の頂点座標データ配列から x 座標を取り出す。取り出された 2 つの頂点データの x 座標値を比べ、原点 x0 から近い頂点座標で吸着処理を行う。

このとき各配列で定義した辺と頂点は、カイト K1(=ピース番号 1 のカイト) を基準として、短い辺は「l」、長い辺は「L」とし、時計回りに番号をつけ、これを辺名とした。頂点にも同じく時計回りに角度番号 (A,B,C,D) をつけ、これを頂点名とした。(図■)

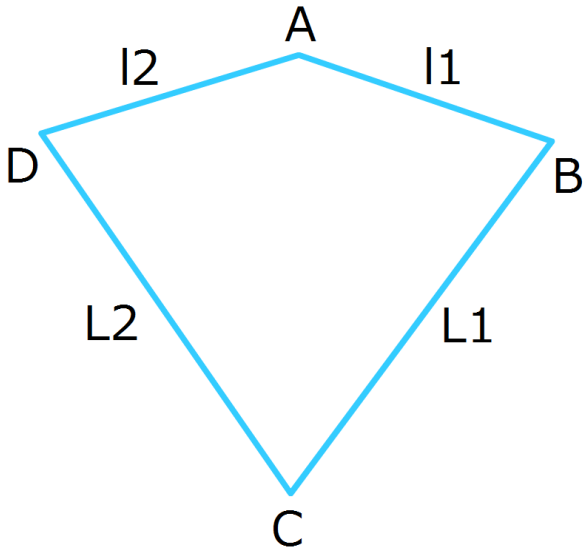

図 33: カイト (K1) の辺名と頂点名

同じくダートの各辺と各頂点にも名前をつけていく。ダートで基準となる形としてはD1(=ピース番号1のダート)を採用した。採用理由としては、K1とすべての辺と頂点がくっつくダートの形がD1であるためである。これからD1はK1のどの辺と頂点がくっつくかを対応づけさけ、下図のように各頂点と辺に名前をつけた。

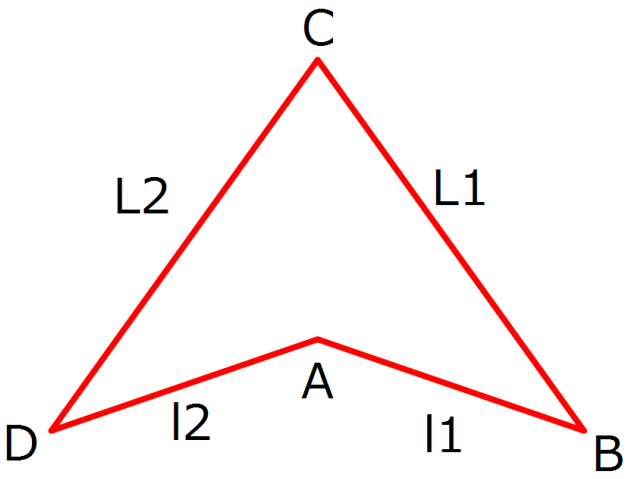

図 34: ダート (D1) の辺名と頂点名

以上のような辺名・頂点名が、K1・D1を基準として各ピースオブジェクトに対応づけられていく。これは、K1・D1以外のオブジェクトはK1・D1を回転させたときに得られる形であるため、回転したとしても定義した辺名・頂点名はかわらないということである。
上記の通りに、各ピースオブジェクトの辺と頂点に名前をつけた一覧が下図である。

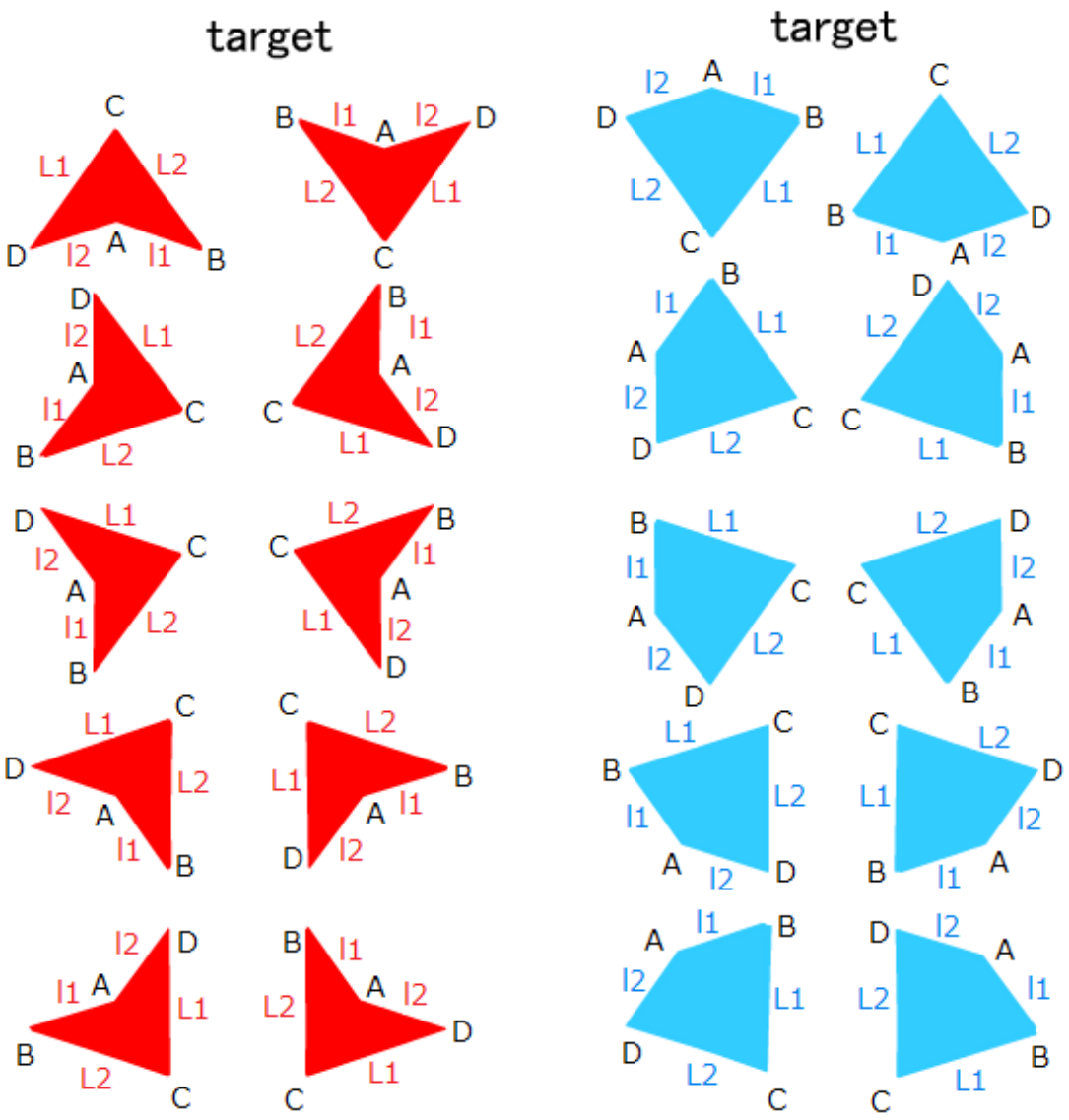

図 35: 各ピースオブジェクトの辺名と頂点名

以上の各ピースオブジェクトの辺名と頂点名を各配列に対応づけさせる。使用した配列は、ピース番号を i としたとき、下記の 3 種類となる。

- 各辺の傾き角度が格納される配列　　k(d)i_line = [l1, L1, L2, l2]
- 各頂点の X 座標の値が格納される配列　k(d)i_pointX = [Ax, Bx, Cx, Dx]
- 各頂点の Y 座標の値が格納される配列　k(d)i_pointY = [Ay, By, Cy, Dy]

まず最初に、辺の傾き角度は事前に計算した値を配列にそれぞれ格納している。格納した値は表 1, 表 2 の通りである。

表 1: カイトピースの各辺の角度値 (i=ピース番号)

ki_line ki	[0] l1	[1] L1	[2] L2	[3] l2
k1	162	54	126	18
k2	162	54	126	18
k3	54	126	18	90
k4	90	162	54	126
k5	90	162	54	126
k6	54	126	18	90
k7	126	18	90	162
k8	18	90	162	54
k9	18	90	162	54
k10	126	18	90	162

表 2: ダートピースの各辺の角度値 (i=ピース番号)

di_line di	[0] l1	[1] L1	[2] L2	[3] l2
d1	162	54	126	18
d2	162	54	126	18
d3	54	18	126	90
d4	90	54	162	126
d5	90	54	162	126
d6	54	18	126	90
d7	126	90	18	162
d8	18	162	90	54
d9	18	162	90	54
d10	126	90	18	162

次に、各頂点座標の x,y 値を各配列に格納していく。この時、各オブジェクピースの頂点座標計算はプログラム内で行い、算出したデータを各配列に格納する。

この時も座標計算基準として、K1,D1 を採用する。まずオブジェクトの基準点を原点 (0,0) として、下図のように各頂点の座標を考える。

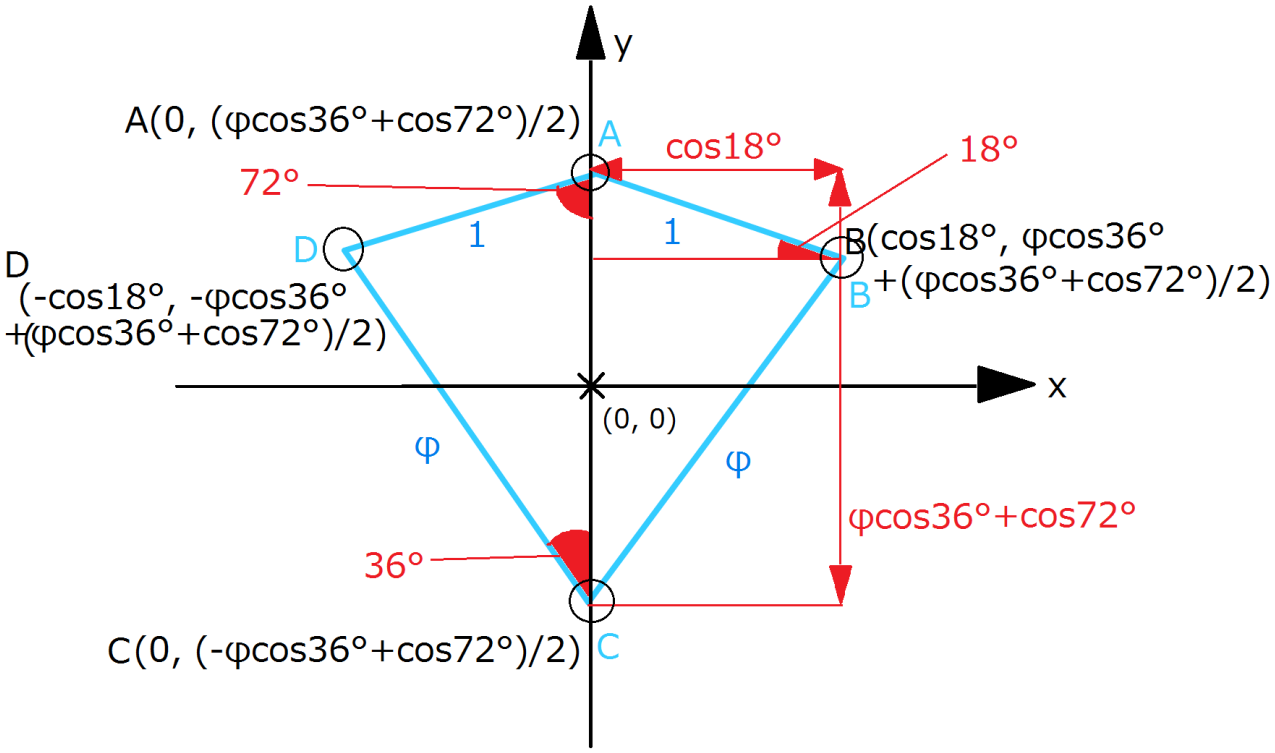

図 36: カイト (K1) の頂点の座標値

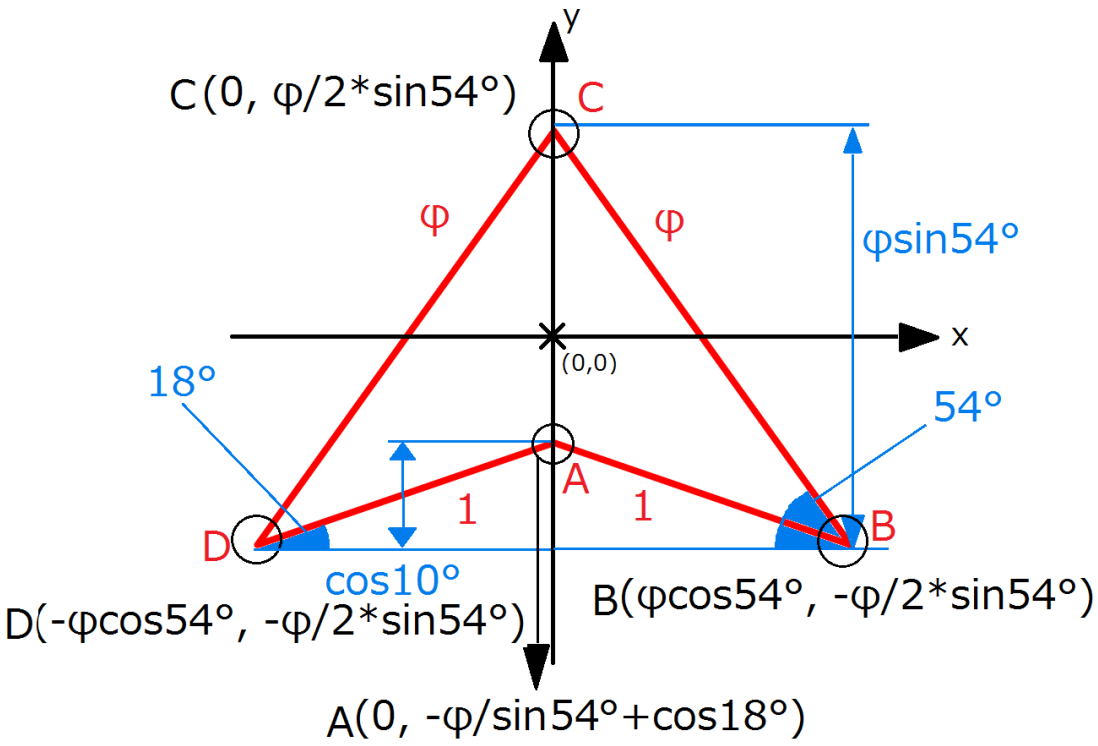

図 37: ダート (D1) の頂点の座標値

この時の座標値は、各辺の値を ϕ (=1.618) と 1 とで算出させているため、実際の値より遥かに小さくなっている。そのため、この座標値に実際のオブジェクトの幅の倍率を掛けなければいけない。オブジェクト幅 (z=カイトの倍率、w=ダートの倍率 として) の倍率は以下の式を使ってプログラム上で算出させる。なお、各オブジェクトの横幅は、カイトオブジェクトの横幅=74、ダートオブジェクトの横幅=60 である。

$$z = \frac{74}{\phi \cos 36^{\circ} + \cos 72^{\circ}} \tag{2}$$

$$w = \frac{60}{\phi \sin 54^{\circ}} \tag{3}$$

以上の式から求められたオブジェクト幅の倍率を各座標値にかけ、配列に格納していく。これで K1 と D1 の頂点座標を求めることができた。他のピースオブジェクト座標は基準オブジェクトを回転した時に得られるため、各オブジェクトの回転角度を事前に算出し、プログラム上で計算させる。回転式は下記の通りである。

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \tag{4}$$

そして事前に算出した各オブジェクトの回転角度データは表3, 表4の通りである。

表 3: カイトピースの各辺の回転角度 (i=ピース番号)

Ki	回転角度
K1	0 °
K2	± 180 °
K3	+72 °
K4	-72 °
K5	+108 °
K6	-108 °
K7	+144 °
K8	-144 °
K9	+36 °
K10	-36 °

表 4: ダートピースの各辺の回転角度 (i=ピース番号)

Ki	回転角度
D1	0 °
D2	± 180 °
D3	-108 °
D4	+108 °
D5	-72 °
D6	+72 °
D7	-36 °
D8	+36 °
D9	-144 °
D10	+144 °

以上の角度・式・座標値を使って算出させた各ピースオブジェクトの座標値が配列に格納させる。これで各配列への格納がすべて終了した。

この配列情報から接触判定を行い、吸着処理も行う。接触判定を行う際には、クリック選択されているピースオブジェクトを「k_new」、接触相手を「k_old」として考えていく。このとき、k_new と k_old と判定されたピースオブジェクトが持つ配列データが k_new と k_old の配列に継承される。なお配列名は k_new が、k_new_line、k_new_pointX、 k_new_pointY。k_old が、k_old_line、k_old_pointX、 k_old_pointY である。

タイリング可能相手であるかどうかの判定として、まず接触し合うオブジェクトの辺の傾き角度が同じであるかを判定させる。この時注意することは、同じ角度であっても k_new と k_old の辺が、l=l、L=L の関係でなければ false になるところである。そのため以下のような判定となる。

表 5: k_new と k_old の辺接触判定

判定式	判定結果
k_old_line[0] == k_new_line[0]	true
k_old_line[0] == k_new_line[1]	false
k_old_line[0] == k_new_line[2]	false
k_old_line[0] == k_new_line[3]	true
k_old_line[1] == k_new_line[0]	false
k_old_line[1] == k_new_line[1]	true
k_old_line[1] == k_new_line[2]	true
k_old_line[1] == k_new_line[3]	false
k_old_line[2] == k_new_line[0]	false
k_old_line[2] == k_new_line[1]	true
k_old_line[2] == k_new_line[2]	true
k_old_line[2] == k_new_line[3]	false
k_old_line[3] == k_new_line[0]	true
k_old_line[3] == k_new_line[1]	false
k_old_line[3] == k_new_line[2]	false
k_old_line[3] == k_new_line[3]	true

しかし上記の判定だけでは、オブジェクトピースが図形の内側から重なる現象や、k_new が近づいてくる方向を考慮していないため、くっつく辺は if 文が最初に true となった辺 1 つだけになる現象が起きてしまう。このような問題があるため、この他にもどの方向からピースが近づいているのかの判定も行う。判定方法は、k_new、k_old の基準点の座標差を求め、左右の位置情報を変数 migi に、上下の位置情報を変数 ue に代入される 0 と 1 で判別する。

表 6 に示したように、座標差の x の値が正の場合 k_new は k_old の右側に位置し (migi=1)、負の場合は左に位置している (migi=0) とみなす。また、y の値が正の場合 k_new は k_old の下に位置し (ue=0)、負の場合は上に位置している (ue=1) とみなす。

表 6: k_new と k_old の辺接触判定

座標差	上下左右	変数=0, 1
X 正	右	migi = 0
X 負	左	migi = 1
Y 正	下	ue = 0
Y 負	上	ue = 1

そして表 5 の接触判定が true の場合、true となった k_new 辺の中点座標 (bunX, bunY) を求める。この中点座標と、表 6 の変数 migi,ue をみて、本当に接触するべき辺であるかの判定を行う。

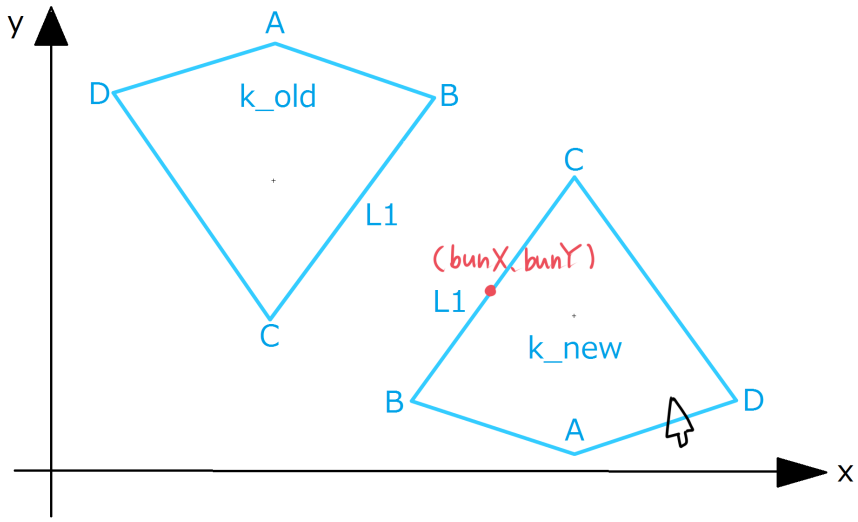

図 38: k_new と k_old の接触判定

bunX, bunY, migi, ue を参照する関数 hontouniyouika を新しく追加し、この中で判定を行う。関数 hontouniyouika 内で変数 hantei(初期値 0) を定義し、return で判定結果の値を返すようにする。この時、hantei=1 の判定は false となり、hantei=0 は true となる。hantei=1(false) になる判定は以下の通りである。

表 7: 関数 hontouniyouika 接触判定

中点 (bunX, bunY)	migi, ue	hantei
bunX 正	migi = 1	1
bunX 負	migi = 0	1
bunY 正	ue = 1	1
bunY 負	ue = 0	1

以上の判定により、ピースが内側から重なることなく、近づく方向によって吸着する辺が決まるようになった。

5　まとめ

結論として、ペンローズタイリングの法則性、規則性を視覚的、感覚的に理解できるパズルを制作することができた。

実際にパズル制作中に気付いたこととして、ペンローズタイリングは20種類のタイルでつくれるということがわかった。この20種類というのは、カイト、ダートのピース番号を足した数である。つまりは、カイトとダートが回転した形も1つのタイルだとみなしたとき、それぞれ10種類のタイルができあがるということである。このことからヒントを得て、今回のパズル制作ではその要素を多く盛り込むことができた。

制作するにあたって1番手のかかった処理は、ピース同士の吸着であった。一般的なジグソーパズルは1つのパズルに対して最低で2つ、最高でも4つの特定のピースしか当てはまらない。しかし、ペンローズタイリングパズルでは、1つのピースに対して何通りもピースが当てはまるパズルであるため、判定処理がいくつも必要になり、思っていた以上の時間がかかった。

また、時間がかかった理由の一つとして、AdobeFlashのASコンパイルがバグを検出しにくいところも挙げられる。そのためトレース文で地道に値を管理する必要があり、見つけにくいミスを見つけるのにもよく時間がかかった。

今後の発展としては、パズルピースの色を変えられるようにし、もっと視覚的にわかりやすく、タイリングしやすいようにする。時間制限を設けたり、点数をつけるなどをしてゲーム要素を盛りこみもっと楽しいものにする。などがあげられる。

参考文献

[1] 「エッシャー・マジック だまし絵の世界を数理で読み解く」,(東京大学出版会,2011/1/5）[p.3-10]

[2] 「Newton 別冊「カタチ」をめぐる数学の不思議図形に強くなる」,(ニュートンプレス,2010/8/10）[p.102-105]

[3] 「別冊サイエンス マーチン・ガードナーの数学ゲーム」,(日本経済新聞出版社,2013/2/22）[p.102-105]

[4] 「デザインのための数学」,(オーム社,2011/12/25）[p.28-34]