数理解析研究所講究録
第1735巻 2011年 148-161

ミニ研究でのルービックキューブ3D表示プログラムのバージョンアップ

福島高専モノづくり教育研究支援センター技術職員　和賀 宗仙
Toshinori Waga
Fukushima National College of Technology,
Manufacturing Support Center for Education and Research

1　はじめに

福島高専では第2学年に，各研究室に学生を配置しグループで半年間研究を行う「ミニ研究」が前期1単位で行われている．第5学年の卒業研究は1年間かけて個々の学生が研究を進めるが，ミニ研究では若干規模の小さい研究テーマで，研究室に配属された数人の学生が共同で研究を進める．一般教科の研究室に配属されることも可能であり，数学科では2×2×2のルービックキューブをテーマにしたミニ研究を行ってきた．数学科島袋研究室では，ルービックキューブの全ての組み合わせを最短手数で分類した電子的なデータベースを作成し，そのデータベースから最短手数による回転公式を見つけ出した．回転公式とは，この手順でキューブを回転するとこの2つのキューブの位置が入れ替わり他のキューブは元通りになるとか，この2つのキューブの向きだけが入れ替わるといった回転の手順である．

井川，西浦研究室ではこれらの公式を用いたルービックキューブの解法をテキストとして作成し，小中学生を対象にした公開講座で用いている．2×2×2のルービックキューブは8個のキューブを持っているが，そのうち上半分の4個のキューブの色を揃える一面完成法，その後8個すべての色を揃える二面完成法の順番で説明している．公開講座では，ルービックキューブを完成させるのに必要ないくつかの回転公式を，担当教員が受講生たちを前にして教壇上でかざすようにしながら見本を見せ，説明していた．受講生には各自2×2×2のルービックキューブが配布されるので，見本を見ながら真似をしていく．しかし，このやり方では教壇上で行われている見本を目で追いつけなかったり，回転公式をどこまで実行したのかを見失うことが多々あり，その都度補助学生が見回って元に戻していた．

そこで，平成21年度の公開講座ではマウスでキューブの回転操作ができるよう，コンピュータの3DCGでルービックキューブを表示することにした．Visual C++ 6.0による3×3×3ルービックキューブの3DCG表示のプログラムは既にインターネット上にあったため[1]，隅の8つのキューブを取り出すことにより2×2×2仕様に改造した．公開講座の場所は通常ゼミ室から情報センターのコンピュータ演習室に移り，回転公式

の説明はスクリーン上にコンピュータ画面を映し出し，マウス操作でキューブ回転する様子を3DCG表示で見せながら行われた．受講生もプログラムを起動させ，マウス操作で回転公式を実行し，混乱した場合はプログラムを再起動することでスムーズにやり直しができ，とても理解しやすくなった．なお，回転公式の説明後は，キューブの向きの定義や群論といった数学的背景について説明し，公開講座であげている回転公式のみで，いかなる崩れた状態のルービックキューブも完成させることが可能であることを説明する．今後もこのような公開講座を行っていくにあたって，その内容に見合った形でのプログラムの改良はとても有用である．

平成22年度は，ミニ研究を通してこのプログラムの改良が行われた．画面上で1クリックするごとに指定された回転公式を自動適用していくようにバージョンアップしよう，というものである．本稿ではその取り組みの経過，作業分担の様子，教育成果について報告する．

2 ルービックキューブ3DCG表示の流れ

ルービックキューブの置かれた空間には，8つのキューブの頂点が同時に接する点を中心(以下ルービックキューブの中心)としたワールド座標が定義されている(図1(a))．キューブがx軸方向に沿って2個，y軸方向に沿って2個，z軸方向に沿って2個並べられ，計8キューブのルービックキューブが構成されている．ここで，ワールド座標とは空間上における絶対的な座標であり，視点座標とは視点を原点とした動く座標である．視点座標の座標軸は図(図1(b))の状態から視点座標をθ回転させ，さらにϕ回転し，原点を視点に平行移動することで決まる．ここで，ワールド座標の原点と視点が一致することはないものとする．PC画面上では，マウスの右ボタンを押しながら左右に動かすとθが変化し，上下に動かすとϕが変化し，ルービックキューブの中心から視点との距離は一定とする．

点(X_f, Y_f, Z_f)をワールド座標系上における視点の座標とすると，ワールド座標系上の点(X, Y, Z)は次式により視点座標系上の点(X_e, Y_e, Z_e)に変換される．

$$\begin{pmatrix} X_e \\ Y_e \\ Z_e \end{pmatrix} = \begin{pmatrix} -\sin\theta & \cos\theta & 0 \\ \sin\phi\cos\theta & \sin\phi\sin\theta & -\cos\phi \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & -\sin\phi \end{pmatrix} \begin{pmatrix} X - X_f \\ Y - Y_f \\ Z - Z_f \end{pmatrix} \tag{1}$$

こうして求められた視点座標系の座標を，投影面上の座標（ディスプレイの座標）に直すには次式を用いる．

$$X'_e = \frac{X_e}{Z_e} \quad , \quad Y'_e = \frac{Y_e}{Z_e} \tag{2}$$

プログラム上では，各キューブの番号を図2(a)(b)のように割り振っている．各キューブにおいても，(c)のように頂点番号，(d)のように面番号が割り振られており，各キューブの中心座標，各キューブの各頂点の座標，各面の中心座標をデータとして管理してい

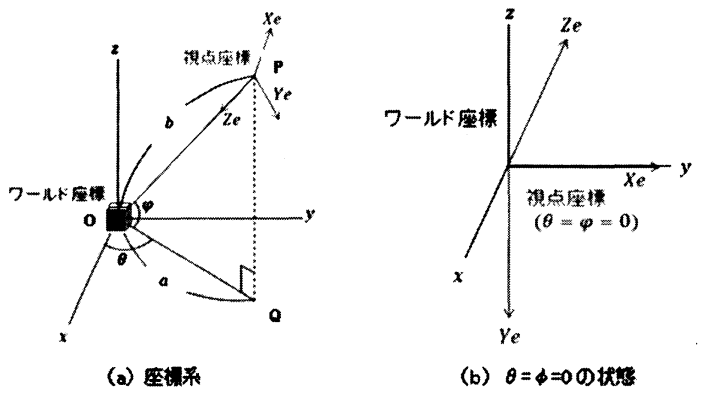


図 1: ワールド座標系と視点座標系の定義

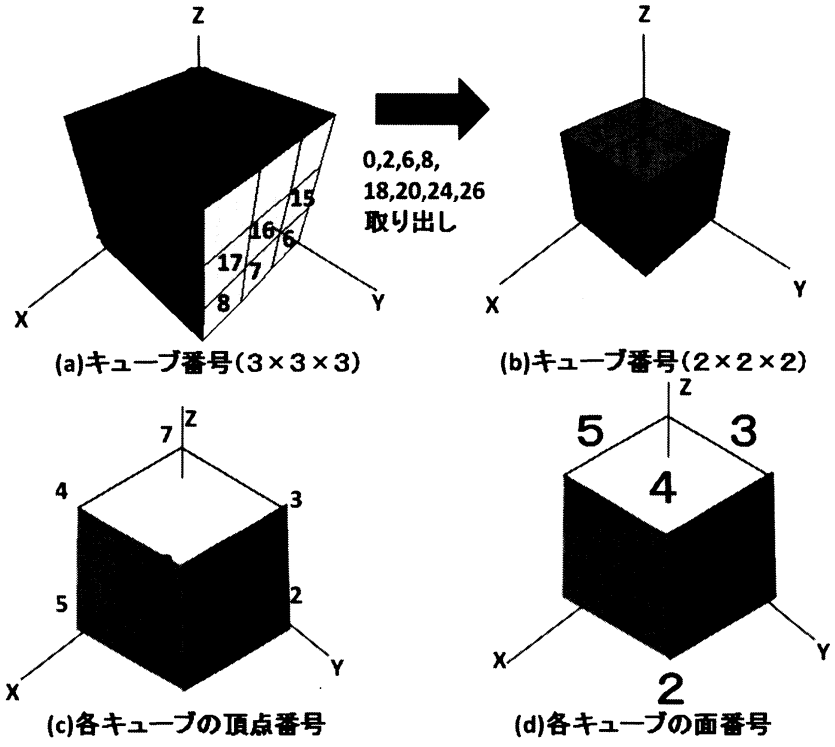


図 2: ルービックキューブにおける番号割り振り

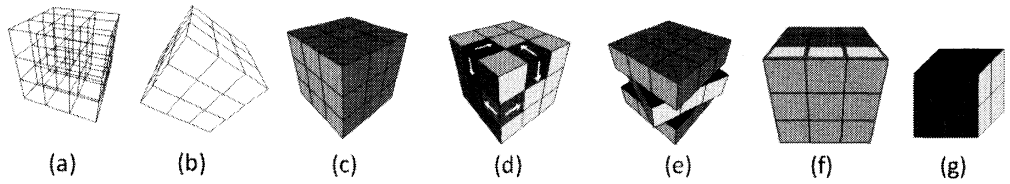


図 3: ルービックキューブ 3DCG 表示生成過程

る．これによって，各キューブの頂点をワールド座標空間上で回転させたり，あるキューブの面と視点との距離を計算するといった命令が可能になっている．

式 (2) により，各キューブの頂点を座標変換しすべての辺をディスプレイ上に描くと図 3(a) のワイヤーフレームモデルによる表示が得られる．

(b) はサーフェイスモデル表示であり，以下のようにして実現する．ルービックキューブを描画する前に，各立方体 (キューブ) の中心と視点との距離を計算し，視点から遠い順に並び替え，その順番で描画する．これは重ね塗り法と呼ばれるものである．各キューブを次の手順で描く．各面に対し，頂点の座標をもとに，面の内側から外側へ向かう法線ベクトルを求めることができる．これに対し，視点から面の中心へ向かうベクトルを視線ベクトルと呼ぶ．法線ベクトルと視線ベクトルが (おおよそ) 向かい合っていれば可視面であり，(おおよそ) 同じ向きならば不可視面となる．可視か不可視かの判断には，法線ベクトルと視線ベクトルの内積を計算し，正ならば不可視面とみなし描画せず，負の面のみを描画する．内積が 0 のときは視線の向きと面が平行なときである．

次に面に色をつけたのが (c) である．面の色並びは配列データで用意している．

キューブの回転は左ドラッグで行うのであるが，まずは左クリックによって面を選択する必要がある．ルービックキューブの各面の各頂点はディスプレイ上の 2 次元座標の値をもっており，面はディスプレイ上には四角形で描かれている．面の各頂点の 2 次元座標の値を用いて，左クリックした点がどの四角形の中に含まれているかを判別することができる．ただし，陰面処理によってディスプレイ上には表示されない面もデータとしては存在しているので，可視面のうち一番視点に近い面を選択面とすればよい．

さて，面を選択したあと，回転の方向は 2 通りある．マウスをドラッグしたとき，ドラッグ方向のベクトルが，選択面の隣り合う 2 辺のどちらに近いかで，回転方向を判断するのだが，それには図 4 のように内積を用いる．回転候補である面の方向ベクトルを E_1, E_2，ドラッグ方向のベクトルを E_3 とする．E_3 と E_1 のなす角を ϕ，E_3 と E_2 のなす角を θ とし，E_2 方向の単位ベクトルと E_3 との内積 $= |E_3| \cos\phi$，E_2 方向の単位ベクトルと E_3 との内積 $= |E_3| \cos\theta$ を比較する．前者のほうが大きければ $\phi < \theta$ であるから，E_1 を回転方向として選び，後者が大きければ E_2 を選ぶ (図 3(d))．

そしてドラッグしたマウスカーソルの移動距離に応じて回転角度を計算し，回転する 9 つのキューブに対して，各頂点の 3 次元座標をアフィン変換し式 (2) でディスプレイ座標に変換して回転途中の状態を表示する．そのときには，再び陰面処理を施すことが必要である (図 3(e))．

ボタンを離したとき，回転角度から，90 度単位でどちらの状態に近いかを判断し，そ

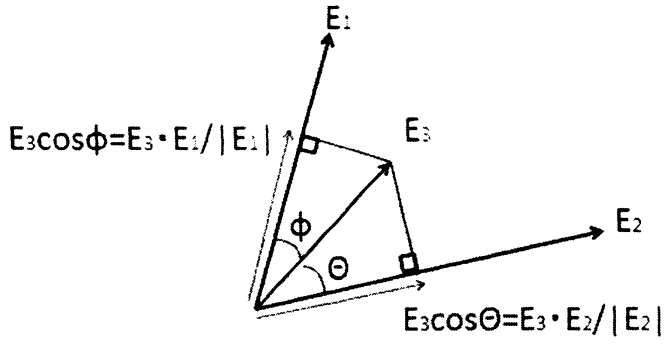


図 4: 内積を用いた回転方向の決定

れに応じて面の色を塗り替えて，各立方体の頂点はドラッグ前の状態に戻すことで回転操作が完成する (図3(f))．面の色の塗り替えには，配列データ (以下面の色塗り替えデータと呼ぶ) を用いる．その配列データは，回転軸 (x 軸，y 軸，z 軸のいずれか) と回転する 9 個のキューブ列を指定して 90 度回転したときに，何番目のキューブがあった位置に何番目のキューブがきて，何番目の面があったところに何番目の面がくるかを示すものである．

福島高専の公開講座で扱っているルービックキューブは $2 \times 2 \times 2$ (図3(g)) なので，図2(a) から隅にある 0,2,6,8,18,20,24,26 番目の合計 8 個のキューブを取り出し，これらにあらためて 0〜7 のキューブ番号を振りなおせばよい．面の色の塗り替えデータも取り出した 8 個のキューブに関するものを取り出して組みなおす (図2(b))．ワールド座標系の原点は，図2(a) では 13 番目のキューブの中心，(b) では 8 つのキューブの一頂点が同時に接する点である．

3 C言語の学習

ルービックキューブの 3DCG プログラムに用いる言語は Visual C++ 6.0 であり，C 言語→ C++→ Visual C++ 6.0 の順番で学習する必要がある．平成 22 年度のミニ研究では計 18 時間をかけてプログラミングの講義をしたが，C 言語の制御文，関数まで教えるのがやっとであった．講義のテキストには [2] を用いた．ミニ研究の最終発表は前期末の 9 月 28 日であり，時間が限られているため，7 月 8 日時点でプログラミング言語の講義を打ち止めとした．パラメータを渡して値を返すという関数の概念までは教えたため，プログラミングはチームで分担可能であることまでは理解してもらえたと思う．しかしながら，全くのプログラミング初心者に 3 週間計 12 時間の講義と自宅学習可能なオンライン教材で C/C++を教える公開講座を開催している大学もあるため [3]，そちらでの講座内容の組み立て方を調べ，次年度からの参考として活かしていきたい．

4 プログラムの改造箇所

改良前のプログラムにおいて，２×２×２のルービックキューブの手を進めるのに重要となる関数は次の２つである．

```
/*============================================================================
 機能　回転すべき立方体４個を回転させる
 引数　point : 現在のマウスの位置
============================================================================*/
void CRubic::RotateCubes( CPoint point )

/*============================================================================
 機能　回転後、面の色を塗り替えて新しい状態にする
 引数　point : 現在のマウスの位置
============================================================================*/
void CRubic::SetNewState( CPoint point )
```

この２つの関数のうち前者は，ルービックキューブの面を選択してドラッグをしているときに回転方向が決定した後に，動かす４個のキューブの各頂点のワールド座標の回転を行う関数である．後者は，ドラッグを終えてマウスのボタンを離したときに，面の色を塗り替えて新しい状態にする関数である．回転はマウスのドラッグによって行われるため，引数には両者とも現在のマウスの位置が使われている．

しかし，回転公式を実行する際には，公式の何手目にどの列(どの４つのキューブ)を，x, y, z のうちどちらの軸を中心に，どちらの方向に回転するかを指定することになる．その公式の各手を格納するための構造体を次のように用意する．

```
/*
================================================
    公式の構造体
================================================
*/
typedef struct stFstep
{
    int  m_AxisCube;     // 回転対象となるキューブの列番号
    Axis m_RotateAxis;   // 回転軸
    int  m_Direction;    // 回転方向
}Fstep;
```

この構造体を用いて，次の２つの関数を追加し，上に挙げた関数に多態性(同じ関数名でも引数の与え方によって処理が異なる)を持たせる．

```
void CRubic::RotateCubes( Fstep step, int d );
void CRubic::SetNewState( Fstep step );
```

stepは，実行する公式の一手を表している．dは回転角度であり，stepに基づいて0〜90までdの値を1ずつ動かし，1°ずつキューブを回転させて90°までゆっくり動かしてからキューブの各頂点の座標を元に戻し，SetNewStateでstepに基づいて面の色を塗り替える．以上の構造体と関数の追加は，ミニ研究学生メンバーのうちソフトウェア研究部員でもある3人(以下プログラム担当メンバー)が担当した．Fstep構造体に与える値と実際のキューブ回転の対応関係は図5のとおりである．

公開講座では，テキスト[4][5]で，特定の2つのキューブの位置を入れ替えたり，特定の2つのキューブの向きを替えたりするための公式を示している．図5を用いて，各公式をFstep構造体の配列データに変換する作業を数学担当メンバーで行い，プログラム担当メンバーに渡し，ソースコードに埋め込んでもらった．一例として，[5]に示す3つのキューブの位置を入れ替える公式3の変換例を図6に示す．8手あるので，Fstep構造体の要素数8の配列となっている．

5 ルービックキューブに関する群論

回転公式には，2つのキューブの向きを変えるものはあっても，1つのキューブの向きだけを変えるものは存在しない．また，ルービックキューブは[4][5]で挙げている回転公式のみですべての状態から元に戻すことが可能である．このことを数学的に理解してもらうため，ミニ研究生には，以下に述べる向きの保存則とルービックキューブ群の構造定理まで教えている．これにより，ルービックキューブの組み合わせの総数が導かれる．以下にはキューブの向きの保存則，ルービックキューブ群の構造定理および，それらにもとづいた今後のプログラム改良の課題について述べる．

5.1 キューブの向き

ルービック・キューブの操作の性質を調べるためにキューブの向きを数値化したものを定義する．まず,全てのキューブが，向きは違っているかもしれないが，正しい位置にあるとする．このとき,各キューブの正しい向きからのずれをν_i(iはキューブの名前)と書いて頂点の周りの反時計回りに120°の回転を単位として表すことにする．即ち,キューブiが正しい向きにあれば$\nu_i = 0$, 正しい向きから反時計回りに120°回転した状態であれば$\nu_i = 1$, 正しい向きから時計回りに120°回転した状態であれば$\nu_i = -1$である．時計回りに360°回転させたものは回転していない状態と同じなので向きに関しては$3 = 0$となる．時計回りに120°回転したものは反時計回りに240°回転させたものを同じだから向きに関しては$2 = -1$となる．つまり，ν_iは$\boldsymbol{Z}_3$に属する．

次に一般の配置にあるキューブの向きを定義する．そのために(頭の中で)ルービック・キューブを二つ用意し,一つは初期状態のままにしておき,もう一つは崩す．二つのルービック・キューブの各キューブの青い面と白い面に(頭の中で)マジックで印をつけてお

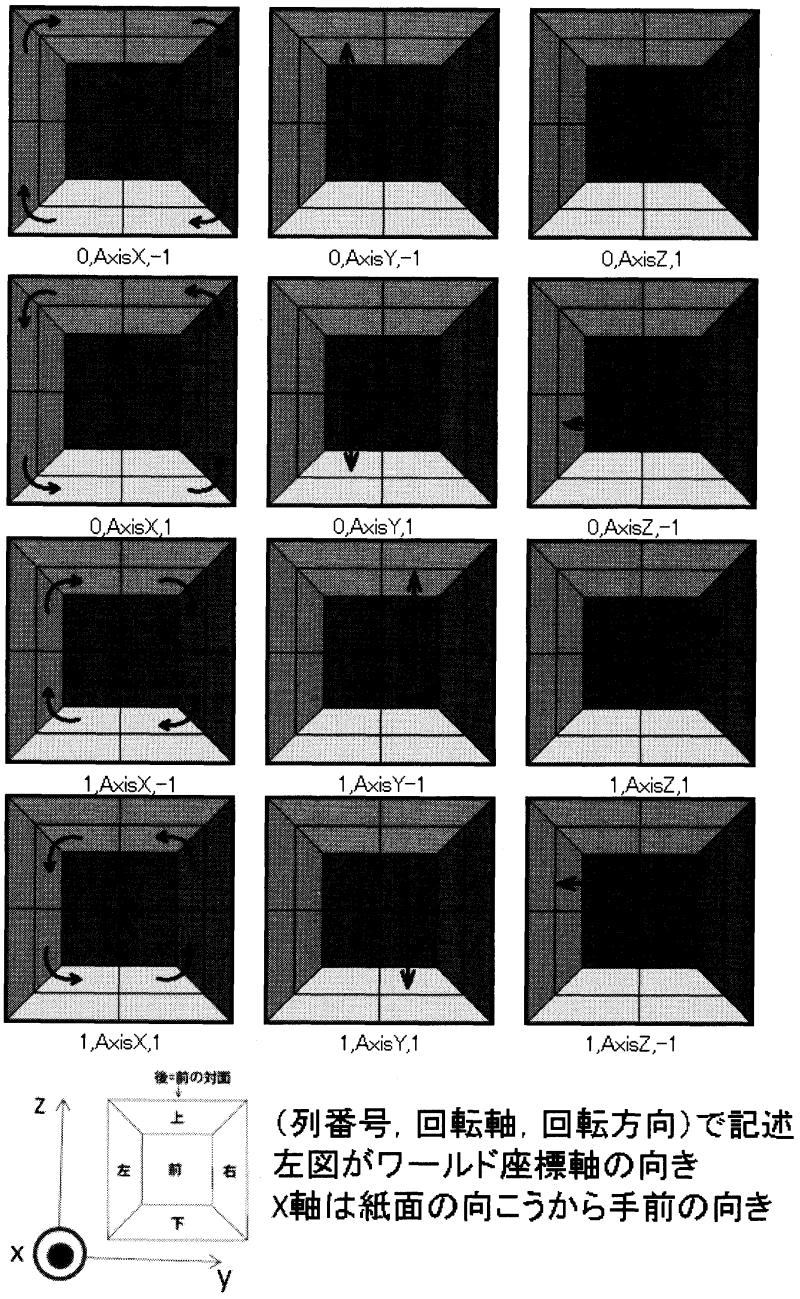
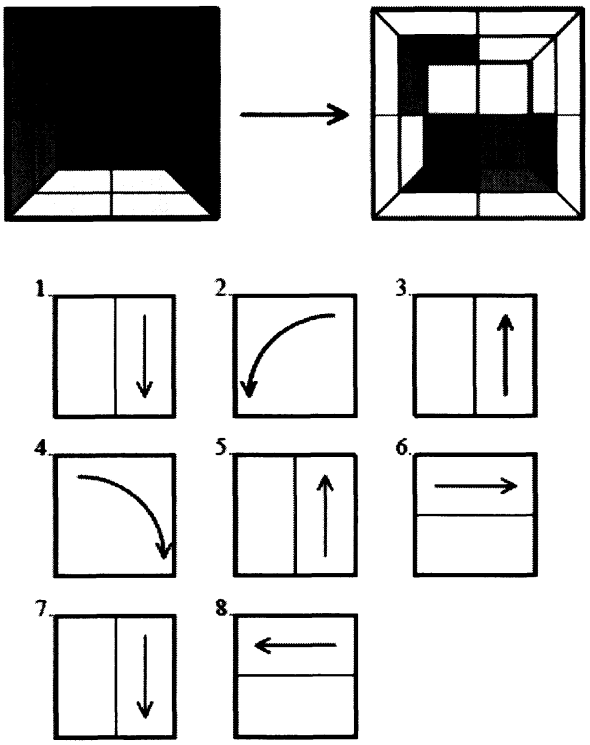


図 5: ルービックキューブの実際の回転と Fstep 構造体との対応

公式 3




```
const Fstep koushiki3[8]={
        {1,AxisY,1},
        {1,AxisX,1},
        {1,AxisY,-1},
        {1,AxisX,-1},
        {1,AxisY,-1},
        {1,AxisZ,1},
        {1,AxisY,1},
        {1,AxisZ,-1}
};
```

図 6: 公式の Fstep 構造体への変換例

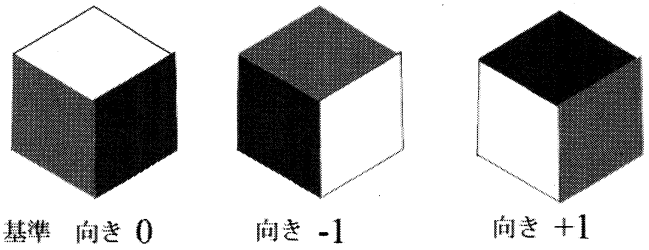


図 7: キューブの向きの定義

く. 初期状態では青い面と白い面は互いに対面の関係にあるので各キューブにつきただ一つの面にのみ印がついたことになる. 二つのルービック・キューブの対応する位置にあるキューブを比べ印のずれを先程のように頂点の周りの 120° の回転を単位にして ν_i と表す. 上及び下面に関する回転に関しては明らかに ν_i は不変である. 左右及び前後面に関する時計及び反時計回りの 90° の回転では二つのキューブの ν_i が $+1$ され残り二つのキューブの ν_i が -1 される. 初期状態では各 $\nu_i = 0$ だから次の保存法則が得られる.

定理 1. [向きの和に関する保存則]([6]) $\boldsymbol{Z}_3$ の世界で 8 個のキューブの向きを全てたすと常に 0 になる： $\sum \nu_i = 0 \bmod 3$.

5.2 $2 \times 2 \times 2$ ルービック・キューブ群

$2 \times 2 \times 2$ のルービック・キューブの操作の全体は操作の合成を積として有限群をなす. これを G_2 と書き $2 \times 2 \times 2$ の**ルービック・キューブ群**と呼ぶ. $\mathfrak{S}_n$ で n 次対称群を表し, 位数 3 の巡回群を $\boldsymbol{Z}_3$ と表す. キューブの位置を $1, \cdots, 8$ で表し, 操作 A によって $1, \cdots, 8$ の位置にあるキューブがそれぞれ $\sigma_A(1), \cdots, \sigma_A(8)$ に移ったとする. このとき,

$$\sigma_A = \begin{pmatrix} 1 & \cdots & 8 \\ \sigma_A(1) & \cdots & \sigma_A(8) \end{pmatrix} \in \mathfrak{S}_8$$

とおく. 次に操作 A によるルービック・キューブの向きの変化を表現することを考える. そのために少し準備をする. $\mathfrak{S}_8$ は $(\boldsymbol{Z})^8$ に次のようにして自然に作用する:

$$\sigma(\nu_1, \cdots, \nu_8) = (\nu_{\sigma^{-1}(1)}, \cdots, \nu_{\sigma^{-1}(8)})$$

このとき, 次の補題が成り立つ.

補題 2. $\sigma, \tau \in \mathfrak{S}_8, \mu \in (\boldsymbol{Z})^8$ に対して

$$(\sigma\tau)\mu = \sigma(\tau\mu).$$

証明 $\mu = (\mu_1, \cdots, \mu_8)$ とおく. このとき,

$$\begin{aligned} (\sigma\tau)\mu &= (\mu_{(\sigma\tau)^{-1}(1)}, \cdots, \mu_{(\sigma\tau)^{-1}(8)}) \\ &= (\mu_{\tau^{-1}\sigma^{-1}(1)}, \cdots, \mu_{\tau^{-1}\sigma^{-1}(8)}) \end{aligned}$$

ここで

$$\begin{aligned}\nu &= (\nu_1, \cdots, \nu_8) \\ &= \tau\mu = (\mu_{\tau^{-1}(1)}, \cdots, \mu_{\tau^{-1}(8)})\end{aligned}$$

とおくと $\nu_i = \mu_{\tau^{-1}(i)}$. よって

$$\begin{aligned}\sigma(\tau\mu) &= \sigma\nu = (\nu_{\sigma^{-1}(1)}, \cdots, \nu_{\sigma^{-1}(8)}) \\ &= (\mu_{\tau^{-1}\sigma^{-1}(1)}, \cdots, \mu_{\tau^{-1}\sigma^{-1}(8)}) \\ &= (\sigma\tau)\mu.\end{aligned}$$

以上で主張が示された.

$(\boldsymbol{Z}_3)^8$ の部分群 $\{(\nu_1, \cdots, \nu_8) \mid \sum \nu_i = 0\}$ は $(\boldsymbol{Z}_3)^7$ と同型なので記号を簡単にするために $(\boldsymbol{Z}_3)^7 = \{(\nu_1, \cdots, \nu_8) \mid \sum \nu_i = 0\}$ とおくと $\mathfrak{S}_8$ の $(\boldsymbol{Z})^8$ への上の作用は $(\boldsymbol{Z}_3)^7$ を不変にする. 操作 A を行う前の位置 i にあるキューブの向きを ν_i と表し, 操作 A を行った後の位置 i にあるキューブの向きを $\nu_{A;i}$ と表し, $\nu_A(i) = \nu_{A;\sigma_A(i)} - \nu_i$ とおくと

$$\nu_A = (\nu_A(1), \cdots, \nu_A(8)) \in (\boldsymbol{Z}_3)^7.$$

$\nu_A(i)$ の定義は次のように言い換えることもできる.

$$\nu_A(i) = \text{「初期状態に操作 } A \text{ を施したとき } i \text{ の位置にあるキューブの向き」}$$

このとき, $A = (\sigma_A, \nu_A) \in \mathfrak{S}_8 \times (\boldsymbol{Z}_3)^7$ と表すことにする操作 A の次に操作 B を行う, この一連の操作を BA と普通の数の積と同じ記号で表す. 数の積の場合と違って AB と BA とは必ずしも等しいとは限らない. 二つの操作 $A = (\sigma_A, \nu_A), B = (\sigma_B, \nu_B)$ に対して BA を $\mathfrak{S}_8 \times (\boldsymbol{Z}_3)^7$ の元として表示してみる.

$$\begin{aligned}\nu_{BA}(i) &= \nu_{BA;\sigma_B\sigma_A(i)} - \nu_i = \nu_{BA;\sigma_B\sigma_A(i)} - \nu_{A;\sigma_A(i)} + \nu_{A;\sigma_A(i)} - \nu_i \\ &= \nu_B(\sigma_A(i)) + \nu_A(i) = (\sigma_A^{-1}\nu_B)(i) + \nu_A(i).\end{aligned}$$

よって

$$(\sigma_B, \nu_B)(\sigma_A, \nu_A) = (\sigma_B\sigma_A, \sigma_A^{-1}\nu_B + \nu_A), \tag{3}$$

$$(\sigma_A, \nu_A)^{-1} = (\sigma_A^{-1}, -\sigma_A\nu_A) \tag{4}$$

が成り立つ. 上の方法で直積 $\mathfrak{S}_8 \times (\boldsymbol{Z}_3)^7$ に群の構造を入れたものを $\mathfrak{S}_8$ と $(\boldsymbol{Z}_3)^7$ の**半直積**といい, $\mathfrak{S}_8 \ltimes (\boldsymbol{Z}_3)^7$ と表す.

定理 3. [ルービック・キューブ群 G_2 の構造定理]

$$G_2 = \mathfrak{S}_8 \ltimes (\boldsymbol{Z}_3)^7.$$

特に,$2 \times 2 \times 2$ のルービック・キューブの組み合わせの総数は

$$\#(G_2) = 8! \cdot 3^7 = 2^7 \cdot 3^9 \cdot 5 \cdot 7 = 89994240.$$

図 8: 平成 22 年度ミニ研究発表写真

8 つのキューブをバラバラに解体した状態から，ルービックキューブを組み立て直すことを考える．8 つのマスにキューブをはめ込むとき，キューブの位置の並びは 8!通りあり，7 個までは 3 通りの向きを自由に選べるが，残りのキューブ 1 個は**定理 1** により向きが自動的に決まることを，式 (3) は意味している．この**定理 3** にもとづき，バラバラにした各キューブを向きを決定しながら 8 つのマスに，マウス操作で配置してルービックキューブの初期状態を決める機能を追加することが，今後のプログラム改良の課題である．

6　今年度のミニ研究発表

ルービックキューブの 3DCG 表示には行列やベクトル解析の数学的知識が使われるが，第 2 学年前期では未習事項となり，ミニ研究で独自に学ぶことが必要であった．発表はポスターセッション形式であり，ミニ研究で勉強した数学知識の内容，公式の変換手順，プログラムの改良手順に 8 人で手分けをして発表した．写真を図 8 に示す．

発表の中で説明した内積と外積は，3DCG 表示のどの過程で使われているのかという質問があり，数学班が一生懸命答えてくれた．プログラム班には，適用する公式を実行中に切り替えられないのかという指摘があった．発表会の時点では，プログラム中にはすべての公式データがソースコードの中に記述されていたが，その中のどれを選ぶかはあらかじめソースコードの中に書き込み，実行中には切り替えられなかった．

その後，キーを押すことによって色配置の初期化と適用公式の切り替えが行われるよう平成 22 年 12 月 1 日時点で機能追加をした．キーの割当ては以下のとおりである．このプログラムは平成 22 年 12 月 4 日（土）および 11 日（土）の公開講座にて使用予定であり，スムーズな進行が期待される．

逃がしのテクニック ： N キー
一面完成
- LV1 パターン 1〜4 ： ZXCV キー
- LV2 パターン 1〜4 ： ASDF キー
- LV3 パターン 1〜6 ： QWERTY キー

二面完成
公式 1〜7 ： 1〜7 キー
色配置の初期化 ： その他のキー

7 まとめと今後の展望

第2学年前期にプログラミングと必要な数学知識を教え，公式を自動実行できるようルービックキューブの3DCG表示プログラムをバージョンアップすることができた．数学的知識もプログラミング知識がまだ乏しいため，臨機応変に作業分担割り当てを考えていく必要があったが，以下のような成果が得られたと思う．

- いかなるWINDOWS言語よりも先にまず，C → C++ → WINDOWSプログラミングの順番で勉強する道標を示せたことは，ソフトウェア研究部学生の今後にも有用であったと考える．
- プログラムが動作するまでのバグ症状の切り分けにプログラム班が粘り強く取り組んだ
- 今後すぐに授業で学ぶベクトル解析，行列について先立って勉強でき，数学を意欲的に学ぶきっかけをつくれた．
- 数学担当班は運動部員でもあり，公式の変換作業を手書きで行う作業に元気に取り組んでくれた．
- ミニ研究で作った電子ファイルはプログラム，発表資料等すべて学内SNSにアップロードしたが，SNSはプログラムのバージョン管理，分担プログラミングにとても役にたつことがわかった．

今後ルービックキューブ教材として使えるようにしていくために，次のようにプログラムを改良していくことを考えている．

- 各キューブの向きを数値で画面表示したい
- ルービックキューブの色配置の初期状態を自動的に乱数生成したりGUI操作で作れるようにしたい
- 適用する公式の選択過程を説明しながら自動で解いていく3D教材の作成
- LEGOブロックで自動的に解くロボットの作成
- ルービックアースなどの他の種類のルービックキューブへの応用

参考文献

[1] http://www.geocities.co.jp/SiliconValley-Bay/4543/Rubic/index.html

[2] 粂井 康孝, “猫でもわかるC言語プログラミング (猫でもわかるプログラミングシリーズ)” ソフトバンククリエイティブ, 2004.

[3] http://www.u-aizu.ac.jp/images/ja/public/openclass/h22/h22try03.pdf

[4] http://www.fukushima-nct.ac.jp/ math/kmki.pdf

[5] http://www.fukushima-nct.ac.jp/ math/RUBIK-H20.pdf

[6] 前田ひろみ・中村孔一共著, Rubik Cube のひとつの解法, 数理科学 (1981.11), pp. 78–83.