3 D技術を用いた立体感覚養成ソフトウエア

Production of a computer software for training the three-dimensional sense

田縁正治

3 DCGと呼ばれる三次元コンピュータグラフィックスを利用して立体感覚を養成するソフトウエアの開発を行った。これを利用すると、立方体を組み合わせてさまざまな立体を作成することで、立体の前後左右および上下の立体の様子を頭に描きながら操作するので、立体感覚を身につけることが期待できる。また、目標を設定するために立体作成メニューを用意した。このメニューには立方体を組み合わせて作成できる完成形を表示した。このメニューからひとつの立体を選択しその立体を作成すると、完成した時に正しく作成できたことのチェックを行うことができる。メニュー項目は初級者用と中級者用の 2 種類を用意し能力の異なる幅広い人が利用できることとした。また、時間を計ることで更に練習意欲を向上できるようにした。作成したソフトウエアはさまざまな人に使用してもらい、そのスピードおよび操作方法が十分に使用に耐えるものであることが確認できた。また、使用してもらった被験者から、このソフトウエアを譲ることを希望される等、楽しみながら立体感覚を養成するためのソフトウエアとして出来上がっていることが確認できた。

キーワード：3 次元コンピュータグラフィックス、ソフトウエア、C++、WindowsXP、バーチャル

目　次

Ⅰ　序論

脳を鍛えるということは最近多くの人の関心を集めている。能力トレーニングと呼ばれている。もっと簡単に脳トレと呼ばれることもある。もちろん単に鍛えるだけならば、読み書きそろばんと呼ばれていたように古くから行われていた。最近の特徴は楽しく鍛えるということであろう。漢字、簡単な計算、反射神経といったさまざまな内容を持つ脳力トレーニングがあり[1]、コンピュータやゲーム機が活躍している。ゲーム機は基本的にコンピュータと変わらないので、コンピュータのみで考察すれば十分である。基本的なトレーニング方法は問題が出されそれに対して答えるという形で行われる。ではなぜコンピュータが利用されるのだろうか。問題が紙媒体で出されるのか、コンピュータで表示されるのかという違いがトレーニングに大きな違いをもたらす。コンピュータを利用することの利点は次の５点にまとめることができる。１．問題をランダムに選択できるので、紙媒体に比べて飽きがこない。２．動きのある問題や答えるスピードを計るなど時間を効果的に利用することができる。３．問題に対して答えを入力すると即座に判定される。４．過去の履歴を保存することができるので、トレーニングの進歩が分かりやすく示されるので意欲を引き出すことが容易にできる。５．単に正解や不正解といった判定のみならず脳年齢など複雑な判定をすることができる。

これまでにさまざまなトレーニングが開発されてきたが[1-3]、まだ立体感覚を効果的に養成するトレーニングは開発されていないように思われる。立体感覚をトレーニングすると、方向音痴の改善や車の運転技術の向上、複雑な機械の設計といったさまざまな発展が期待される。そのためか、公務員や民間企業などの採用試験でも出題されることがある。立体感覚を効果的に養成する方法を開発することは意義があると思われる。

本研究では、３Ｄ技術を用いたコンピュータソフトウエアを開発することで立体感覚を楽しく養成することを目指した。開発に当ってはソフトウエアのスピードが十分であることと、その操作方法が複雑にならないことを目標とした。３Ｄを利用したソフトウエアでは通常の２Ｄを利用したソフトウエアと比べて処理量が多い。陰線処理、斜めになった物体の座標計算、複数の物体を表示する場合のＺオーダーの処理など、２Ｄでは必要がなかった処理が多いため、稚拙な開発方法では十分なスピードを実現することができない。その結果ぎこちない動きのソフトウエアとなりとても楽しくトレーニングすることにはならない。また、操作方法が複雑でも同じように楽しくトレーニングすることとは程遠いソフトウエアとなる。そこで、これまでに蓄積してきた３Ｄ技術を基礎にしてスピードと簡単な操作を実現するソフトウエア開発を目指すこととした[4-6]。

Ⅱ　開発環境

開発環境は、ＯＳとしてWindowsXPを、グラフィックチップの制御のためにDirectX8.0を利用

した。DirectXの最新版はDirectX9.0cである。しかし、これは利用しなかった。その理由は開発するコンピュータだけで動作するソフトウエアではなく、他のコンピュータでも動作するソフトウエアとしたかったからである。とはいえ、DirectXのバージョンが8.0より低いと機能が低過ぎることになるのでこれは止めることとした。

言語はC++を使用することとし、統合開発環境はMicroSoft社のVisualC++6.0を利用することとした。この理由はDirectXのバージョンと同じ理由で、最新のDotNetFrameworkを利用しない方が望ましいと考えたからである。また、最新のWindowsであるVistaはその機能を参考とするに留めることにした。

Ⅲ　開発の目的

立体感覚を養成する最も良い方法は実際に部品となる小さい立方体を自分の生身の手でさまざまな方法で組み合わせてみることであろう。いろいろと動かしてみながら作成することで立体感覚が養われると思われる。実物を使うので違和感もないであろう。しかし、この方法にも欠点がある。まず立方体を数多く用意しなければならない。また、組み合わせた際に、その組み合わせ方法によっては糊付けしなければその形を保てない場合も多いであろう。そうすると糊付けした後で糊付けした部分を離すような変更をしたい場合には操作がやっかいなことになる。これらの操作は、積み木で遊ぶようなものである。積み木の場合、重力に逆らうような組み合わせはできないことを思い出せば良いだろう。実物で練習する場合は、組み合わせに限界がある。

コンピュータソフトウエアを利用すれば、立方体を用意する必要がない。また、糊付けはコンピュータだから容易に行われる。糊付けした部分をあとで剥がすことも容易である。また、問題が与えられてそれを目標に図形を作成したときに、正誤判定を行うのはコンピュータを利用した方が便利である。このようにコンピュータを使う利点は多い。わかりやすい表示方法と簡単な操作方法を用意すればコンピュータの方が良い場合もあると考えられる。このような考えから立体感覚を養成するコンピュータソフトウエアを作成することとした。

Ⅳ　開発

１．立方体の作成

立体感覚を養成するために部品となる小さい立方体を組み合わせた立体を作成することとしたが、そのために個々の部品となる立方体を作成する方法を決定しなければならない。ここでは、初めから立方体をオブジェクトとして作成しておき、そのオブジェクトを表示するかしないかを決定することで立方体の表示を制御することとした。オブジェクトであるから当然、さまざまな属性を付加することができる。今回は各面の色情報、位置情報、向きの情報に加えて表示・非表示の

情報を付加することとした。

　部品となる立方体は全部で125個用意することとした。初めは中心の１個の立方体のみを表示することとし、左右・上下・前後に最大２個ずつ増加できることとした。つまり、中心に１個、左に２個、右に２個となり、最大５個並ぶ。上下、前後も同様に最大５個並ぶこととした。その結果最大で125個の立方体が表示状態となる。もちろんこの状態では中心の立方体は見えないこととなる。また、部品となる立方体を削除する機能も用意することとした。これは前述の立方体の属性の表示を非表示にすれば良い。

　立方体のデータはまずその頂点のデータを指定するための構造体を作成することから始めた。

```
struct BUILDVERTEX
{
        D3DXVECTOR3 p;      // Vertex position
        DWORD       color;  // Vertex color
        FLOAT       tu, tv; // Vertex texture coordinates
};
```

　構造体の名前がBUILDVERTEXとなっているのは、本研究がこれまでの研究の成果を基にしているためである。ｐは頂点座標、colorは色、tu, tvはテクスチャ座標を表わしている。DirectXの表示モードは次の様に指定した。

```
#define D3DFVF_BUILDVERTEX (D3DFVF_XYZ|D3DFVF_DIFFUSE|D3DFVF_TEX1)
```

　立体の向きを変える操作を行った時は、常に中心の立方体が位置を変えないこととした。中心の立方体は回転するのみでその重心は常にウインドウの中央に位置することとした。中心の立方体に張り付く形で増加した立方体は中心の立方体が回転することに合わせて回転と移動の両方を行う。こうすることで、立体は常にウインドウの中に表示されることが保証される。

　操作を行っていると立方体の向きが分からなくなる可能性がある。そこで、立方体の各面に違う色を塗ることとした。こうすることで、色をみただけで直感的に立体の向きを確認することができる。

　小さい立方体は

```
struct CUBE
{
        BUILDVERTEX v[36];
        float CX, CY, CZ;
        bool Visible;
};
```

とCUBE構造体を作成し、この構造体で表現することとした。36個の要素を持つBUILDVERTEX型の配列v[36]は立方体を構成する頂点の情報を保存するために用意した。ひとつの立方体は６つの面を持つが、ひとつの面は正方形であり。正方形は２つの三角形で表現することができる。ひとつの三角形は３つの頂点をもつ。したがって２つの三角形で表現される正方形が６個ということになり全部で36個の頂点が必要となる。一方この構造体はCX,CY,CZと３つのfloat型の変数をもっている。これは３Ｄで表現したときにこの小さい立方体がｘｙｚ軸方向、何番目の立方体であるかを表している。これにより自分の位置をすばやく知ることができる。最後のbool型の変数Visibleはこの立方体が見える状態なのか、見えない状態なのかを表している。

```
void MakeCube(CUBE& m_CubeTemp,  float X, float Y, float Z );
```

というメソッドを用意し、これを呼ぶことで小さい立方体を作成する。実際のメソッドは以下の通りである。

```
void CMyD3DApplication::MakeCube(CUBE& m_CubeTemp,  float CX, float CY, float CZ )
{
        m_CubeTemp.CX=CX; m_CubeTemp.CY=CY; m_CubeTemp.CZ=CZ;
        //手前
        m_CubeTemp.v[0].p   = D3DXVECTOR3(-dx,dy,-dz);
        m_CubeTemp.v[1].p   = D3DXVECTOR3( dx, dy,-dz);
        m_CubeTemp.v[2].p   = D3DXVECTOR3( dx,-dy,-dz);
        m_CubeTemp.v[3].p   = D3DXVECTOR3( dx,-dy,-dz);
        m_CubeTemp.v[4].p   = D3DXVECTOR3(-dx,-dy,-dz);
        m_CubeTemp.v[5].p   = D3DXVECTOR3(-dx, dy,-dz);
        //左
        m_CubeTemp.v[6].p   = D3DXVECTOR3(-dx, dy, dz);
        m_CubeTemp.v[7].p   = D3DXVECTOR3(-dx, dy,-dz);
        m_CubeTemp.v[8].p   = D3DXVECTOR3(-dx,-dy,-dz);
        m_CubeTemp.v[9].p   = D3DXVECTOR3(-dx,-dy,-dz);
        m_CubeTemp.v[10].p = D3DXVECTOR3(-dx,-dy, dz);
        m_CubeTemp.v[11].p = D3DXVECTOR3(-dx, dy, dz);

        //向こう
        m_CubeTemp.v[12].p = D3DXVECTOR3( dx, dy, dz);
        m_CubeTemp.v[13].p = D3DXVECTOR3(-dx, dy, dz);
        m_CubeTemp.v[14].p = D3DXVECTOR3(-dx,-dy, dz);
        m_CubeTemp.v[15].p = D3DXVECTOR3(-dx,-dy, dz);
        m_CubeTemp.v[16].p = D3DXVECTOR3( dx,-dy, dz);
        m_CubeTemp.v[17].p = D3DXVECTOR3( dx, dy, dz);
```

```
//右
m_CubeTemp.v[18].p = D3DXVECTOR3( dx, dy,-dz);
m_CubeTemp.v[19].p = D3DXVECTOR3( dx, dy, dz);
m_CubeTemp.v[20].p = D3DXVECTOR3( dx,-dy, dz);
m_CubeTemp.v[21].p = D3DXVECTOR3( dx,-dy, dz);
m_CubeTemp.v[22].p = D3DXVECTOR3( dx,-dy,-dz);
m_CubeTemp.v[23].p = D3DXVECTOR3( dx, dy,-dz);
//上
m_CubeTemp.v[24].p = D3DXVECTOR3(-dx, dy, dz);
m_CubeTemp.v[25].p = D3DXVECTOR3( dx, dy, dz);
m_CubeTemp.v[26].p = D3DXVECTOR3( dx, dy,-dz);
m_CubeTemp.v[27].p = D3DXVECTOR3( dx, dy,-dz);
m_CubeTemp.v[28].p = D3DXVECTOR3(-dx, dy,-dz);
m_CubeTemp.v[29].p = D3DXVECTOR3(-dx, dy, dz);
//下
m_CubeTemp.v[30].p = D3DXVECTOR3( dx,-dy,-dz);
m_CubeTemp.v[31].p = D3DXVECTOR3( dx,-dy, dz);
m_CubeTemp.v[32].p = D3DXVECTOR3(-dx,-dy, dz);
m_CubeTemp.v[33].p = D3DXVECTOR3(-dx,-dy, dz);
m_CubeTemp.v[34].p = D3DXVECTOR3(-dx,-dy,-dz);
m_CubeTemp.v[35].p = D3DXVECTOR3( dx,-dy,-dz);
for(DWORD i=0; i<6; i++)
        m_CubeTemp.v[i].color=D3DCOLOR_RGBA(10,200,10,255);
for( i=6; i<12; i++)
        m_CubeTemp.v[i].color=D3DCOLOR_RGBA(200,10,10,255);
for( i=12; i<18; i++)
        m_CubeTemp.v[i].color=D3DCOLOR_RGBA(100,100,150,255);
for( i=18; i<24; i++)
        m_CubeTemp.v[i].color=D3DCOLOR_RGBA(100,150,150,255);
for( i=24; i<30; i++)
        m_CubeTemp.v[i].color=D3DCOLOR_RGBA(150,150,100,255);
for( i=30; i<36; i++)
        m_CubeTemp.v[i].color=D3DCOLOR_RGBA(150,100,150,255);
```

```
        for(i=0; i<36; i++)
                m_CubeTemp.v[i].p += D3DXVECTOR3(CX, CY, CZ);
        for(i=0; i<36; i++){
                m_CubeTemp.v[i].tu=0.0f;
                m_CubeTemp.v[i].tv=0.0f;
        }
}
```

ひとつの正方形の面は２つの三角形で作成している。頂点はv[0]からv[2]でひとつの三角形、v[3]からv[5]でひとつの三角形を作成し、この２つの三角形でひとつの正方形の面を作成している。座標は、これらの頂点のメンバpで表現している。色指定はこれらの頂点のメンバcolorに情報を指定することで表現している。v[0]からv[5]の頂点に同じ色を指定することでひとつの正方形の面が同じ色として表示される。

初めに表示する中心の一個の立方体は

```
MakeCube(m_Cube[0],   0,   0,   CenterZ);
```

として作成した。中心からのずれはCX，CY，CZで表現した。したがって、座標pはD3DXVECTOR3（CX, CY, CZ）だけ位置を移動することとした。

２．操作方法

立体は部品となる立方体を組み合わせて作成することとしたが、立体に対する操作は向きを変えることのみとした。つまり回転をするだけである。こうすることで立体は常にウインドウの中に表示される。この回転操作は全部で３種類用意することとした。左右の回転、上下の回転と手前にある面が常に手前にある状態で回転する３種類である。この回転の操作をマウス操作で実現しようとすると左右の回転はマウスを左右に動かすことで対応することとし、上下の回転はマウスを上下に動かすことで対応づけると、もうひとつの回転の操作に対応するマウスの動きがないこととなる。そこで、ウインドウの左端にボードにみたてた長方形の領域を用意し、その中で上下にマウスを動かすことを最後の回転に対応させることとした。

部品となる小さい立方体を増やす操作はまずどこに増やすのかを指定する必要があった。そこで、現在存在する立方体のひとつの面を指定することで、その面に張り付いた新しい立方体を増やすようにすることとした。こうすれば直感的に位置を指定することが容易である。また、現在指定している面が簡単に確認できることも重要であると考えた。そこで、指定している面の色をあたかも光っているような色に変更することとした。具体的には立方体のひとつの面は２つの三角形をつなげて表示しているので、各三角形の頂点の色を白色に変更することでこの光った感じを表現することとした。一見してどの面が選択状態にあるか分かりやすい状態に仕上がったと思われる。今後同様のソフトウエアを作成する際には参考になると思われる。

３．問題作成と正誤判定

単に立体を作成できるソフトウエアを提供しても、目的がなければ散漫な練習で終わってしまう。目標となる立体を用意して、その作成を目指すことで練習が有効に行われると考えた。そこで、6つの問題を用意した。その中の3つは簡単な問題とし、初心者が練習しやすいように考慮した。残りの3つは初心者段階を終えた中級者を対象とした。

問題作成においては、初心者用と中級者用で少し考え方を変えることとした。初心者用の問題を作成する際は、立方体の側面に平行な変化、つまり縦と横の変化で立方体を増やしていくことで完成するように留意した。したがって、中心にある立方体から、上下、左右、奥行きと手前のみの変化が第1問であり、第2問は中心にある立方体から左右の横の変化で作成できる長い立体を初めに作成し、同様の長い立体を上下に積み上げていく形とした。第3問は上下、左右、前後の変化のみでできる最大の立体を作成することとした。一方、中級者レベルでは、中心の立体が不要で、2つの部分に分かれた立体が第1問であり、第2問は捩れの位置にある細長い立体を作成することを要求し、用意した中で最も難しい第3問は、すべて斜めの位置にある立方体を組み合わせなければできない立体とした。

これらの問題の提示方法は、新しいウインドウを表示し、その中に完成図を表示することで示すこととした。この方法だと、簡単な立体の場合は問題なく立体の形を伝えることができるが、複雑な立体では十分に伝えることが困難であることが分かった。そこで、実際に完成した立体を手本として３Ｄ表示で示す機能を作成することとした。この機能は問題を選んだ直後に見ることができることはもちろんだが、その問題に挑戦してある程度完成に近づいた段階で、現在作成している立体のデータはそのまま保持しておき、一時的に手本となる完成図を見ることもできる方が、利便性が高いと考えた。この機能を提供するには、現在の位置情報をすべて配列に保持しておくと同時に、手本の位置情報を別の配列に保持しておき、メニューから選択した方のデータを３Ｄ表示で利用するように切り替えることとした。このようにすることで、作成中の立体も、手本となる立体も前回回転したままの状態であるので、表示を相互に切り替えても違和感なく操作することができた。

具体的な方法は、問題のデータはint型の2次元配列を用意してデータを格納した。格納するデータは小さい立方体に番号をつけ完成した立体で必要となる番号をデータとして格納することとした。したがって、そのコードは次のようになった。問題ごとに最後に-1というデータを入れることで個数を省略した。

```
int Problem[6][125]={
            {12,37,52,57,60,61,62,63,64,67,72,87,112,-1},
            {0,1,2,3,4,5,6,7,8,9,30,31,32,33,34,35,36,37,38,39,60,61,62,63,64,65,66,67,68,69,90,91,92,9
3,94,95,96,97,98,99,120,121,122,123,124,-1},
            {0,1,2,3,4,5,9,10,14,15,19,20,21,22,23,24,25,29,45,49,50,54,70,74,75,79,95,99,100,101,102,
103,104,105,109,110,114,115,119,120,121,122,123,124,-1},
```

```
            {10,11,12,13,14,35,50,51,52,53,54,59,60,64,69,70,71,72,73,74,85,110,111,112,113,114,-1},
            {4,9,14,19,24,45,46,47,48,49,50,55,60,65,70,75,76,77,78,79,104,109,114,119,124,-1},
            {0,6,12,18,24,26,48,52,72,78,96,104,108,112,116,120,-1}
};
```

所要時間は

```
int StartTimer=0;
```

とスタートの時間を記憶するための変数を用意し、

```
CurrentTime = DXUtil_Timer( TIMER_GETABSOLUTETIME );
ElapsedTime = CurrentTime - StartTime;
if(ElapsedTime>=EndTime) StartTimer=0;
```

と、DXUtil_Timer関数で現在時刻を取得し、StartTimeとの差を計算して経過時間ElapsedTimeを計算する方法を採用した。

手本の表示は

```
int ShowingTehon=0;//0:手本を表示していない　1：手本を表示している。
```

とShowingTehonというint型の変数を用意し、0の時は表示していない、1の時は表示していると約束した。

４．ランキング

練習の意欲を向上するために、中級者では、時間を測定することとした。リアルタイムで時間を表示することと、完成した際に所要時間を示す2通りの時間表示を行うこととした。これによって単純な練習ではなく、時間を見ながら行うのでより意欲の向上が見込まれる。また、時間を記録し、過去の例を基にしてランキングを表示することとした。これによって更に練習の意欲が増すと思われる。ランキングは、中級者用の3つの問題のそれぞれで示すこととした。それぞれの記録はその記録を出した人が自分の名前やニックネームを登録することができることとした。この結果、他の人の結果を目標にがんばることができる。また、記録はひとつの問題に対して5つだけとしたので、がんばって他の人の記録を抜いて自分の記録のみがランキングに入るように努力することも可能である。その他に、自分の過去の記録を塗り替えるようがんばったりすることもできる。過去の自分より良い記録を出すことで自分の練習成果が上がったことを実感できることだろう。このソフトウエアを使用し始めたばかりのころは、当然過去の記録がまったくない状態である。上記のような仕掛けをすることで意欲を刺激するために、あらかじめ初期値を用意しておくこととした。つまり、5つの記録を用意し、それを目標にすることを促す仕掛けをすることとした。この記録はそれぞれ1分、2分、3分、4分、5分として試用に供してみた。それぞれの名前はレベル１、レベル２、レベル３、レベル４、レベル５とした。実際に小中学生が挑戦している様子をみた結果5分というレベルは簡単にクリアできるが、１分は少々骨が折れるようであった。

しかし、１分以内にクリアすることは練習さえすれば決して無理な設定ではないことが分かった。したがって、これらの記録を初期値として採用することとした。

実際に使用に供してみた結果、時間を設定したことで意欲を引き出すことに効果があった。まず、集中力を出すことである。単に回数をこなすだけでなく、より速く完成することを目指すようになった。また、意外に思ったことは、複雑な図形では、まず小さい立方体を可能な限り作成し、その後不要な小さい立方体を削除する方法で完成を目指す人がいたことである。これは当初予想していなかったので、感心したのだが、所要時間をみるとやはり、初めから立体感覚をフルに発揮して作成する方が速く完成するので、より立体感覚の養成に役立つことが分かった。

ランキングを用意するための実際のコードはまず、

```
int RankMark1[6]={60, 120, 180, 240, 300, 0};
char RankName1[6][100]={"レベル1","レベル2","レベル3","レベル4","レベル5", ""};
```

と、ひとつの問題につき２種類の配列を用意した。

RankMark1のデータ60、120等は初期値となる時間データを秒単位で表したものである。ひとつの問題が完成したときはIElapsedTimeに所要時間を代入しておき、次の処理を行ってランキングを更新することとした。

```
for(i=4; i>=0; i--){
        if(RankMark1[i]>IElapsedTime){
                RankMark1[i+1]=RankMark1[i];
                RankMark1[i] = IElapsedTime;
                RankingChanged=1;
                strcpy(RankName1[i+1],RankName1[i]);
                strcpy(RankName1[i] , "あなた");
                RankingChanged=1;
        }
}
```

Ⅴ　今後の開発環境

Windowsの最新バージョンであるWindows Vistaでは、ディスプレイに表示する方法が変更になりエアロと呼ばれる３Ｄによる表示方法が通常の方法として採用されることとなった[7]。しかし、その利用はまだ十分な段階に達しているとはいい難い状態で、３Ｄ効果を利用してみたと呼ぶのがふさわしい程度の利用である。現在のコンピュータのグラフィックス環境を考慮にいれるとこの選択はやむを得ないことかも知れない。しかし、今後３Ｄグラフィックス環境が向上するともっとＯＳレベルで利用されるようになることだろう。

本研究では、一歩リードしてアプリケーション段階で３Ｄ環境を最大限に利用している。３Ｄ技術をＯＳレベルで利用することとなったのは本研究成果を普及させるために助けになると思われる。WindowsVistaでは、３Ｄの他に透明色を利用することも特徴となっている。２つのウインドウが重なったときに前面にあるウインドウのタイトルバーが半透明であるために後ろにあるウインドウの情報を一部見ることができる。この斬新と見える手法は実は以前から利用していたのだが、一般に知られていた訳ではない。そこで、今回のソフトウエアではこの半透明色を利用することで、本ソフトウエアをより新鮮なソフトウエアと見えるように工夫した。今後はさらにWindowsVistaの機能を意識した開発環境を用意することが必要であろう。

Ⅵ　開発結果

開発した結果を大学生、高校生、中学生、小学生に使ってもらった。それぞれ好評で楽しみながら練習するという点に関しては十分な効果があることが分かった。立体感覚については十分な検証を行う方法を用意できなかったので、今回は報告することができないが、学生の中に３Ｄを利用したソフトウエアを開発しそれを基にして卒業論文を書くことを選択した人が２名登場し、影響力のあるソフトウエアとなったことは間違いないようである。

謝　辞

本研究において、宮崎学術振興財団より一部資金援助をいただいた。ここに謝意を表する。

参考文献

１．川島隆太、2006年、「脳を鍛えるパソコンドリル」、インプレス
２．太城敬良、2002年、「右能力がグングンＵＰするマジカル・アイ」、宝島社
３．有限会社ラケータ、2007年、「脳が活性化する！パソコンでできる能力トレーニング」、株式会社ローカス
４．田縁正治、2006年、「３ＤＣＧによる建造物を表すソフトウエアの作成」、宮崎公立大学人文学部紀要、第14巻、第１号、169－179ページ
５．田縁正治、2002年、「建造物の３次元コンピュータグラフィックス」、宮崎公立大学人文学部紀要、第10巻、第１号、113－128ページ
６．田縁正治、2001年、「３ＤＣＧによる大学案内ソフトウエアの作成」、宮崎公立大学人文学部紀要、第９巻、第１号、43－58ページ
７．PCJapan　Vol 11 No 8 p95 2006