３Ｄ脳力トレーニングソフトウエアの開発

Development of a Software for 3D Ability Training

田　縁　正　治

　コンピュータグラフィックスを利用して迷路を３Ｄで表現して、３Ｄ脳力を養成するソフトウエアを開発した。迷路は２種類作成した。ひとつは通常の壁と床で構成される迷路を移動する。もうひとつは迷路の上空からみたときに壁が文字に見えるような迷路である。この２種類の迷路を利用して３Ｄの能力を高めることを目指した。実際に利用してもらうと楽しみながら練習するソフトウエアとなったことが分かった。小学生から大人まで楽しめるので能力を高めるために幅広く利用してもらえそうである。また、Win32APIを利用したのでWindowsはWindows2000からWindowsVistaまでどのOSでも動作した。

キーワード：３D、能力トレーニング、迷路、Win32API、DirectX

目　次

Ⅰ　序論

　最近は自分を鍛えるということがはやっている。鍛えるというとまず思い浮かぶのが体を鍛えるということだが、メタボリックシンドロームといってどうも生活習慣が悪いと脅されるのが最

近の傾向である。肥満症や高血圧、高脂血症、糖尿病から内臓脂肪型肥満と進みさまざまな病気が引き起こされやすくなっている。

一方、脳の方も認知症などを防ぐために鍛えることが話題になっている[1-3]。川島隆太[1]によると「物忘れが多い」、「人の名前が思い出せない」、「会話中に言葉が出てこない」、「脳の老化を防ぎたい」といった現象が認知症に近づくことのようである。対策としてパソコンドリルをしなさいと薦めている。また、「集中力を高めたい」「記憶力を高めたい」「人付き合いがうまくなりたい」といった理由で脳を鍛えたいという人もいる。川島の著書では平面的なドリルのみが紹介されているが、立体的な物を上手に認識する必要があることがある。たとえば、立体物のデザイン、迷路の移動、車の運転、各種就職試験などさまざまである。そこで、本研究では立体的な物の認識能力を高めるための方法を開発することを目的とした。

立体的な感覚が必要になる状況は主に２つ考えられる。ひとつは、Ａ：複雑な立体を目の前で動かすことである。もうひとつはＢ：迷路のような立体の中を自ら動き回ることである。このような状況をうまく乗り切るにはそれぞれの能力を養成する必要がある。立体感覚を養成するには実際に複雑な立体物を操作したり、迷路の中を動き回ったりすることが考えられる。これらの方法は有効であると思われるが、さまざまな立体物を用意したり、迷路の中を移動して回ったりする必要がある。経済的あるいは時間的な制約を受けている場合は以上の方法はあまり実用的ではない。一方、複雑な立体物や迷路をコンピュータソフトウエアとして作成すれば、上記の問題点を解消する良い方法となる可能性を秘めている。本研究は、上記の第２のＢ場合について実際にコンピュータソフトウエアを作成する方法を研究することとした。

迷路を迷わずに移動する能力を養成するために次の３つの方法を用意することとした。１．実際に迷路を通る。２．迷路の構造を理解する。３．迷路を自作する。この考えを基に、迷路の中を移動するコンピュータソフトウエアは、次の３つの機能を用意することとした。１．いくつかの迷路をあらかじめ用意しその中を動き回って目的地まで達する。２．立体を上からみると文字となるように作成する。この場合文字は立体物を構成する壁の組み合わせで作成されている。上からみるとなんという文字かすぐ分かるが、壁のすぐ横を移動するとなんという文字か分からない。しかし、壁の周りを移動すると文字を判別することができる。３．迷路を自分で作成する。

以上の試みを行い実際にテストすることとした。

II　ソフトウエア開発環境

ソフトウエアを開発する言語は、COBOLやJavaやBASICが考えられる。また、Webアプリケーションとするならば、PHPも視野に入るであろう。しかし、今回は３Ｄで表示するソフトウエアを開発するので、コンピュータの性能を十分に発揮できる言語を使用することが重要であると考えた。このため、グラフィックボードの利用を前提とし、より効率の良いグラフィック命令を利用する必要がある。この方法として、OpenGLが考えられるが、今回は使用しなかった。

今回の作成はVisualC++とDirectXの組合せで行った。また、Win32APIを使用しWindows2000からWindowsVistaとさまざまなOSで動作可能であることを目標とした。操作方法はキーボードの矢印キーを使用して前後左右に迷路の中を移動することとした。この方法は、以前利用した方法でキーボードに慣れた人には使いやすい方法である[4-6]。また、時間を計ることとして能力の向上を実感できることとした。

III　ソフトウエア開発

１．迷路の大きさ

本研究では小学生でも利用できるような迷路を３Ｄとして作成して利用することとした。そこで、迷路の大きさを決定することから始めた。図１は公務員試験でだされそうな問題である。

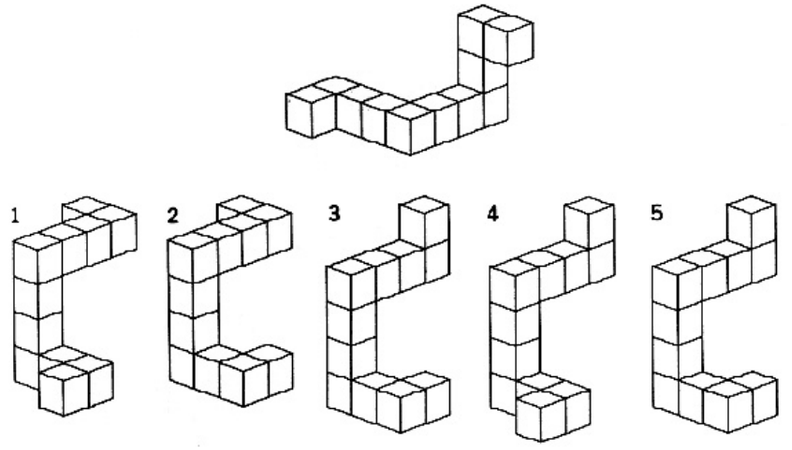

図１　立方体を組み合わせた立体

問題の内容は上の図と同じ立体は、１から５の立体図のどれかを問うものである。上の立体図と下の５つの図は向きが違う。この５つの図の中には上の図と比較したときに向きが違うだけでまったく同じ図がひとつだけある。立体感覚がない人から見るとどれが同じものかなかなか分からないだろう。この程度の問題でも公務員試験などでは役にたつので、いたずらに大きい立体で練習する必要はないと考えられる。そこで、縦10、横10の正方形を組み合わせた迷路を作成することとした。

２．迷路の作成

迷路はイメージとしては縦10、横10の正方形が図２のように並んでいると考えると良い。この図が床の上に広げられており、ひとつの正方形の上に壁が建設されることがあったり、床のままだったりする。壁があれば当然突き抜けて壁の向こうに行くことはできないように設定した。一方床のままの場合は先に進むことができるように設定した。

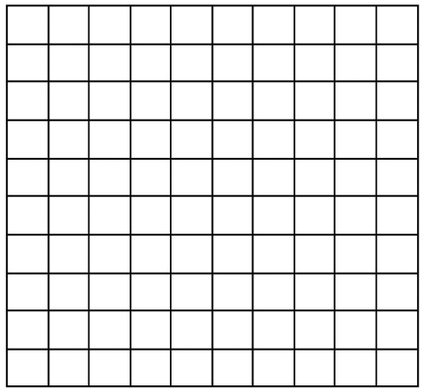

図２　迷路作成のイメージ図

迷路は２種類作成することとした。ひとつは初めに迷路の入り口に位置していて、そこを出発点として前後左右に移動することにより迷路の出口を目指すという、いわゆる単純な迷路である。

もうひとつは、作成方法は単純な迷路と同様に縦10、横10の正方形が図２のように並んでいてそれぞれの正方形が床であったり壁であったりするが、迷路を上からみたら、壁の並びが文字に見えるように壁を配置する迷路である。

ひとつの壁は立方体とし、３Ｄグラフィックスによる描き方として採用される３角形を組み合わせる方法をここでも使った。したがって、ひとつの壁の面は正方形となるので、頂点が６個必要になる。この頂点は

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

と構造体を用意して表現することにした。最初のメンバは位置座標、２番目のメンバは色情報、３番目のメンバはテキスチャを貼るときのために用意した。ひとつの壁は立方体だから次のように


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

構造体を用意して表現することにした。最初のメンバは各頂点の中心からの相対位置、２番目のメンバは中心の位置、最後のメンバはこの壁を表示するか否かを示す。

実際に作成する場合は次のような関数を利用した。

```
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);
途中省略
    m_CubeTemp.v[i].p + = D3DXVECTOR3(CX, CY, CZ);
    for( i=0; i<4; i++){
        m_CubeTemp.v[i*6].tu = 0.0f;
        m_CubeTemp.v[i*6].tv = 0.0f;
        m_CubeTemp.v[i*6+1].tu = 1.0f;
        m_CubeTemp.v[i*6+1].tv = 0.0f;
        m_CubeTemp.v[i*6+2].tu = 1.0f;
        m_CubeTemp.v[i*6+2].tv = 1.0f;
        m_CubeTemp.v[i*6+3].tu = 1.0f;
        m_CubeTemp.v[i*6+3].tv = 1.0f;
        m_CubeTemp.v[i*6+4].tu = 0.0f;
        m_CubeTemp.v[i*6+4].tv = 1.0f;
        m_CubeTemp.v[i*6+5].tu = 0.0f;
        m_CubeTemp.v[i*6+5].tv = 0.0f;
```

```
        }
}
```

実際の描画は次の関数で行った。

```
HRESULT CMyD3DApplication::Stage1()
{
   float DRotateX=0.f, DRotateY=0.f, DRotateZ=0.f;
   static int Change=0;
   DWORD i;
   if(StartTimer==1){
        CurrentTime = DXUtil_Timer( TIMER_GETABSOLUTETIME );
        ElapsedTime = CurrentTime - StartTime;
        if(ElapsedTime>=EndTime) StartTimer=0;
   }

   // Set up the cursor
   POINT ptCursor;
   GetCursorPos( &ptCursor );
   ScreenToClient( m_hWnd, &ptCursor );
   m_pd3dDevice->SetCursorPosition( ptCursor.x, ptCursor.y, 0L );
   m_pd3dDevice->SetRenderState(D3DRS_CULLMODE,D3DCULL_CCW);
   m_pd3dDevice->SetVertexShader( D3DFVF_BUILDVERTEX );
   D3DXMATRIX mat, matx, maty;
   D3DXMatrixIdentity( &mat );

   m_pd3dDevice->DrawPrimitiveUP(D3DPT_TRIANGLEFAN,2,m_floor1.v,sizeof
                (BUILDVERTEX));
   m_pd3dDevice->SetTexture( 0, Texture[0]);

   for( i=0; i<NCube; i++)
     if(m_Cube[i].Visible)
          m_pd3dDevice->DrawPrimitiveUP(D3DPT_TRIANGLELIST,12,m_Cube
                [i].v,sizeof(BUILDVERTEX));
   m_pd3dDevice->DrawPrimitiveUP(D3DPT_TRIANGLELIST,12,m_Cube[100].v,
                sizeof(BUILDVERTEX));
```



```
   D3DXMatrixIdentity( &mat );
   m_pd3dDevice->SetTransform( D3DTS_WORLD, &mat );
   m_pd3dDevice->SetTexture( 0, NULL );

   return S_OK;
}
```

壁は現実感が出るように実在の壁を写真撮影してテキスチャとして利用した。移動は１回のキー操作でひとつの正方形の大きさだけ移動することとした。また、移動の他に位置はそのままで向きだけ変わる操作も可能とした。これにより横に移動するときはあらかじめ向きだけ移動しようとする方に変えることでこれから進もうとする方にある壁を確認することができる。回転はキーを１度押すごとに向きを一定の角度だけ変えることとした。初めは１度のキー操作で90度ずつ回転することとしたが、実際に操作してみると、不便であることが分かった。回転が完了するとそれまでに向いていた方の景色がほとんど見えなくなるので、１回のキー操作をするとそれまでとはまったく違った景色が見えるようになりとても戸惑うこととなる。そこで、１回の回転は45度回転するようにしてみた。その結果１度のキー操作を行った結果それまで見えていた景色が半分残っているので操作感が格段に向上した。

３．単純な迷路の作成

迷路は正方形の床を考えたときにこの床が縦10、横10並んでいて、それぞれが通路になるか、壁になるように設計した。イメージを図として図３に示した。灰色の正方形の上には壁があり、白い正方形は単に床があるだけである。Ａは入り口、Ｂは出口である。途中の壁がなければＡからＢまでまっすぐに進めば良い。途中に壁があるせいで迷路となっている。

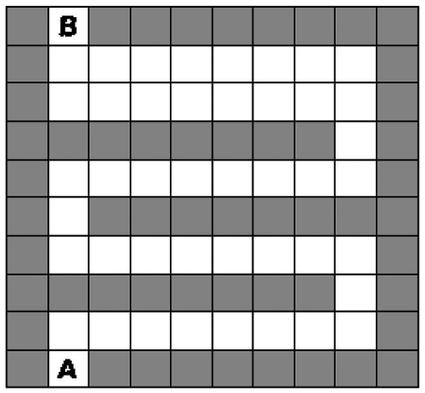

図３　単純な迷路のイメージ図

迷路は全部で６種類用意した。それぞれメニュー項目として、「れい」、「とてもやさしい」、「やさしい」、「ふつう」、「むずかしい」、「とてもむずかしい」とした。これらの迷路では壁のデータは次のようにint型の配列の初期値として用意した。

```
int Problem[6][100] ={
        {0,2,3,4,5,6,7,8,9,10,12,19,20,22,29,30,32,39,40,42,49,50,52,59,60,62,69,7
0,72,73,74,75,76,77,78,79,80,90,91,92,93,94,95,96,97,98,99,-1},
        {0,2,3,4,5,6,7,8,9,10,12,16,20,22,24,26,28,29,30,32,34,36,38,39,40,42,44,4
6,48,49,50,52,54,56,58,59,60,62,64,66,68,69,70,72,74,76,78,79,80,84,88,89,90,91,92,9
3,94,95,96,97,98,99,-1},
        {0,2,3,4,5,6,7,8,9,10,19,20,21,23,25,27,29,30,39,40,42,44,46,48,49,50,59,6
0,61,63,65,67,69,70,79,80,82,84,86,88,89,90,92,93,94,95,96,97,98,99,-1},
        {0,2,3,4,5,6,7,9,10,15,17,19,20,22,24,27,29,30,32,36,37,39,40,44,47,49,50,
52,55,57,59,60,62,63,67,69,70,73,74,75,76,77,79,80,81,89,90,91,92,93,94,95,96,97,98,
99,-1},
        {0,2,3,4,5,6,7,8,9,10,14,19,20,21,22,24,25,26,27,29,30,34,39,40,42,46,47,4
9,50,52,53,54,56,59,60,64,66,68,69,70,71,72,74,79,80,85,86,87,89,90,91,92,93,94,95,9
6,97,99,-1},
        {0,2,3,4,5,6,7,8,9,10,15,19,20,21,23,27,29,30,35,39,40,42,44,46,48,49,50,5
5,59,60,61,63,65,67,69,72,74,79,80,86,88,89,90,91,92,93,94,95,96,97,98,99,-1}
        };
```

図４に実際に作成した迷路の例を示した。高い壁がありその中を移動していく様子が理解できる。

４．文字の形の迷路の作成

文字の形の迷路は、迷路の上空に飛んで迷路を見下ろしたときに、迷路の壁が文字となるような迷路を作成することとした。これにより、単に迷路を移動して立体感覚を養成する他に、クイズ形式で楽しみながら感覚を養成することができる。この場合の迷路のイメージ図を図５に示した。

この図では５という文字を迷路で表現している。必要なデータはint型の配列で次のように用意した。

```
int probCharacter[10][100]={{1,1,1,0,0,0,0,1,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,
1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,
1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,1,1,0,0,0,0,1,1,1},
```


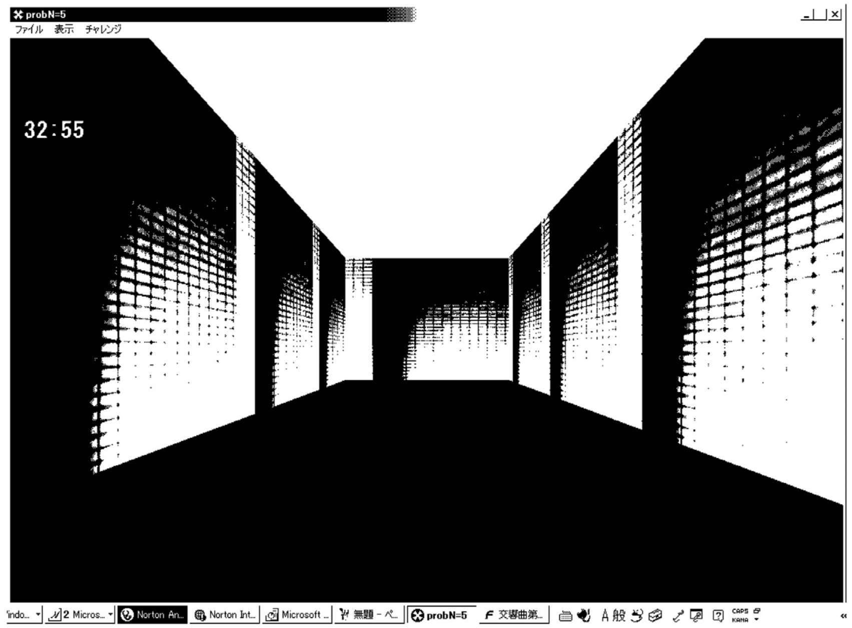

図４　迷路の例

```
途中省略　次のデータは５という文字を表わす
   {1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,
1,1,1,1,1,1,1,1,1,0,1,1,1,1,0,0,0,0,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,
0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1},
途中省略
}
```

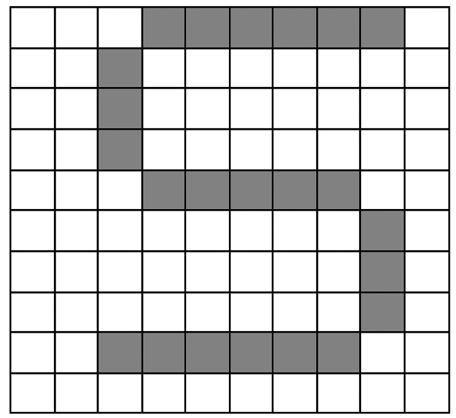

図５　文字を利用した迷路のイメージ図

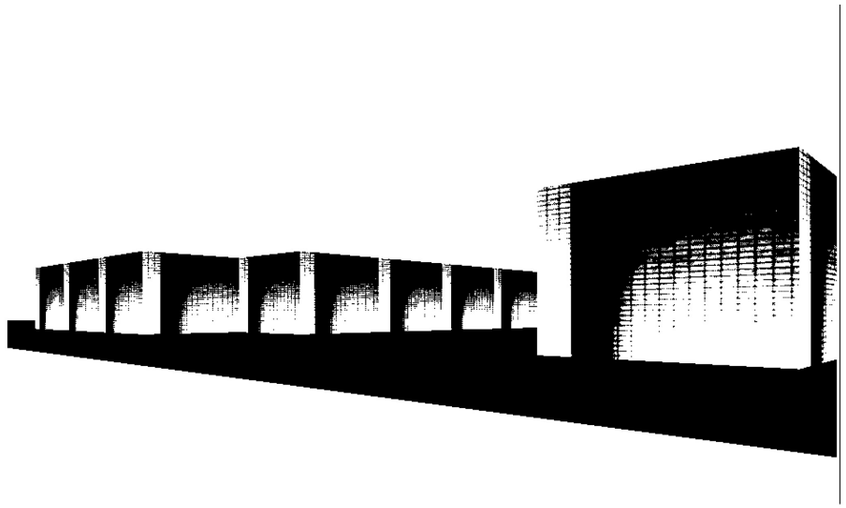

図６　文字を利用した迷路の例

図６に実際に作成した迷路の例を示した。壁が意味ありげにならんでいる様子が理解できる。

壁が文字を表しているので、壁の周りを移動してなんという文字であるのか考える。地上を移動しながら上からみた図を考えるという立体感覚が必要な操作である。文字が分かった場合は最後に確認することができる。図６の例では５という数字であるが、５と正しく答えた場合にはその旨の判定結果を表示するためのウインドウが表示される。図７がその例である。

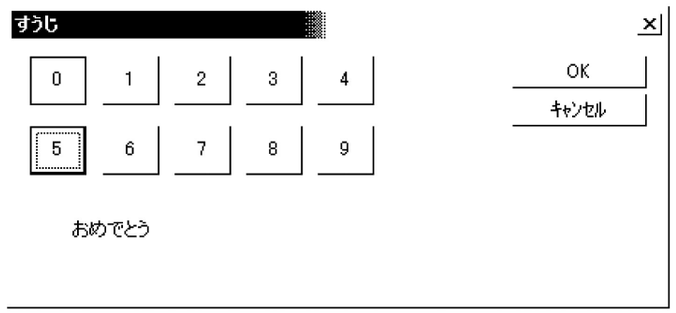

図７　文字の判定結果

Ⅳ　作成したソフトウエアの移植性

作成したソフトウエアはなるべく多くの人が利用しやすいようにすることを目指した。しかし、一方で３Ｄは三角関数を利用した計算や陰線処理などのＺバッファの処理など多くの処理を

行うので、ハードウエアに近いソフトウエアにすることが必要である。このためにJavaVMを介するようなJavaを避けた。したがって、完成したソフトウエアはWindows限定である。Windowsもこれまでにさまざまなバージョンが出された。Windows95、Windows98、WindowsMe、WindowsNTなどは現在でも利用されているかもしれないが、手元にないことと、それらのOSが搭載されたコンピュータは処理速度が遅いと考えられるので、グラフィックスの機能も貧弱であると考えられる。今回は次の３種類のOSで動作確認を行った。Windows2000とWindowsXPとWindowsVistaである。

Windows2000を搭載したコンピュータは自作でCPUはインテル社製のPentium4の１GHzで動作する。

Ⅴ　今後の課題

実際に作成して高校生と大学生に利用してもらった。その結果、想像以上に好評で楽しみながら立体感覚を養成することができることが分かった。好評であった理由のひとつは壁の模様にあるようである。今回は大学の壁の写真を撮影しそれを利用した。その結果、親しみがわき楽しく練習することができたものと思われる。したがって、各人が親しみを持てる写真を用意することで多くの人が楽しく利用できるソフトウエアとなると期待できる。

今後は単に迷路のように３Ｄ脳力を養成するだけでなく、実用的な方面に向かうことを検討してみた。例えば、車の運転の技術向上のためのソフトウエアなど３Ｄの感覚をさらに高めるためのソフトウエアの開発を行う予定である。

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