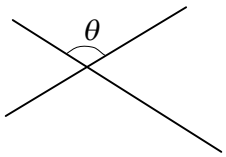
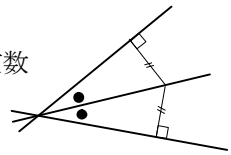
平面解析幾何の基礎事項



１． ２直線　$ax+by+c=0$，　$px+qy+r=0$　が与えられたとき

(1)二直線のなす角度　$\tan\theta=\dfrac{aq-pb}{ap+bq}$　特に直交するとき、$ap+bq=0$



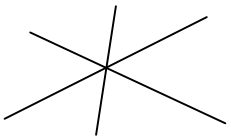
(2)二直線の交点を通る直線の式：$ax+by+c+k(px+qy+r)=0$，k は定数

(3)二直線の交角を二等分する直線　$\dfrac{ax+by+c}{\sqrt{a^2+b^2}}=\pm\dfrac{px+qy+r}{\sqrt{p^2+q^2}}$

(4)一点(x_0,y_0)から直線$ax+by+c=0$までの距離 h は　$h=\dfrac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$

(5)二直線の交点に別の直線　$\ell x+my+n=0$が通る条件は　$\begin{vmatrix} a & b & c \\ \ell & m & n \\ p & q & r \end{vmatrix}=0$

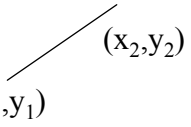
（三直線が一点で交わる条件）



２．２点$\vec{p}_1=(x_1,y_1)$，$\vec{p}_2=(x_2,y_2)$が与えられたとき

(1)この二点を通る直線の式は、　$(x_2-x_1)(y-y_1)=(y_2-y_1)(x-x_1)$

または、$\begin{vmatrix} x & y & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix}=0$



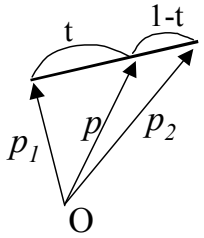
(2)ベクトル$\vec{p}=(x,y)$とパラメータ t を用いた表現では、　$\vec{p}=\vec{p}_1(1-t)+\vec{p}_2t=\vec{p}_1+t(\vec{p}_2-\vec{p}_1)$

とくに、$0\le t\le 1$ のとき、二点を端点とする線分を表す

また、直線が$\vec{p}=\vec{p}_1+t\cdot\vec{n}$のとき、$\vec{n}$が単位ベクトルならば、パラメータ t は距離を表す

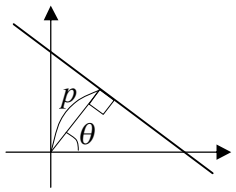
(3)二点$\vec{p}_1$，$\vec{p}_2$をm：nに内分する点$\vec{p}$は　$\vec{p}=\dfrac{n\vec{p}_1+m\vec{p}_2}{n+m}$

または、$\vec{p}_1$，$\vec{p}_2$を t：(1−t)：に内分する点$\vec{p}$は　$\vec{p}=(1-t)\ \vec{p}_1+t\vec{p}_2$



○　直線の Hesse の標準形

$x\cos\theta+y\sin\theta=p$

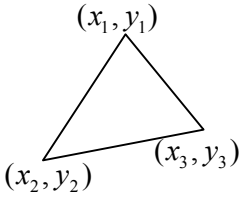


３．３点(x_1,y_1)，(x_2,y_2)，(x_3,y_3)が与えられたとき

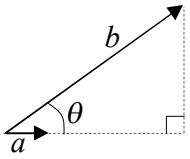
(1)三点で構成される三角形の面積 S は

特に同一直線上にあるときは S=0

$S=\pm\dfrac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$



４．２つのベクトル$\vec{a}=(a_1,a_2)$，$\vec{b}=(b_1,b_2)$の内積$\vec{a}\cdot\vec{b}$

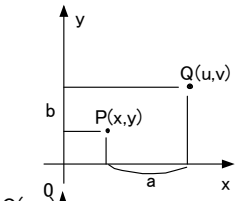


$\vec{a}\cdot\vec{b}=|a|\cdot|b|\cos\theta=a_1b_1+a_2b_2$　　ただし、θは２つのベクトルのなす角度

$\vec{a}$が単位ベクトル（$|\vec{a}|=1$）のとき、内積$\vec{a}\cdot\vec{b}$は、$\vec{b}$を$\vec{a}$方向に投影したときの長さとなる。

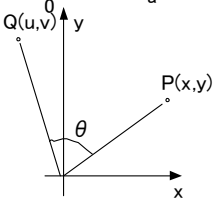
２次元座標変換

２次元平面上の点 P(x,y)を座標変換してできた新しい点を Q(u,v)とすると、座標変換する変換行列は次のように同次座標系を使って表わされる。



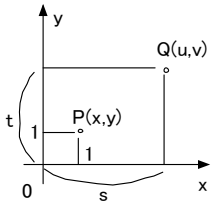
平行移動(並進)（ｘ方向にａ，ｙ方向にｂだけ平行移動させる変換）

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \text{.....(1)}$$



回転移動（原点まわりに角度θ反時計回りに回転移動させる変換）

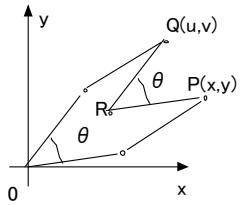
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \text{.....(2)}$$



拡大縮小(伸縮)（原点を中心にｘ方向にｓ倍，ｙ方向にｔ倍する変換）

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} s & 0 & 0 \\ 0 & t & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \text{.....(3)}$$

上記３つの変換を組み合わせて任意の座標変換を行うことができる。



点 R(p,q)の周りに点 P(x,y)をθ回転させて点 Q(u,v)に移動させるとき、

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -p \\ 0 & 1 & -q \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \text{.....(4)}$$

（注）連続した変換を行う場合、変換の順序が重要（行列の積は可換でないから）

一般に、$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ の変換を２次元アフィン変換（Affine transformation）と言う。アフィイン変換は http://www.coins.tsukuba.ac.jp/~fukui/CG/samples/A3.htm で試せる。

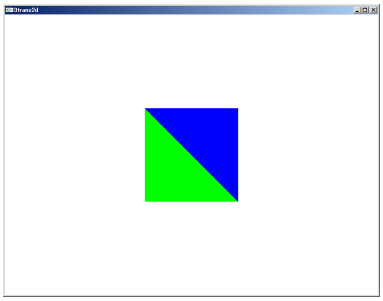
同次座標系の有用性

１．平行移動というベクトルの加算が行列の乗算に置き換えられるため、回転移動も同一の行列の積で表現でき、統一的に変換手続きが行える。

２．一般に同次座標値(x,y,w)を使うことで、中心投影座標を表現できる。無限大点を定義できる。

```
/*   ２次元座標変換   trans2D.c
   http://www.coins.tsukuba.ac.jp/~fukui/CG/ex3.htm を参考
   にして、座標変換を使ったオリジナルの動画を作成すること
*/

#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>

#define    SQUARE 1
float ytrans=0., xtrans=0., rote=-360.; // 表示形状移動用
int       move=0;                       // 自動回転フラグ

void myinit(void)
{ int i;
   glClearColor(1.,1.,1.,0.);  // 消去色
   glClear(GL_COLOR_BUFFER_BIT); // 消去
// 四角形の表示リスト作成開始
    glNewList( SQUARE, GL_COMPILE ); // 形状表示リスト
      glBegin(GL_TRIANGLES); // 三角形定義開始
        glColor3f(0.,1.,0.); glVertex2i(-1,1);
         glVertex2i(-1,-1); glVertex2i(1,-1);
        glColor3f(0.,0.,1.); glVertex2i(-1,1);
         glVertex2i(1,-1); glVertex2i(1,1);
      glEnd( );              // 四角形定義終了
   glEndList( );  // 表示リスト作成終了
}

void reshape(int w, int h)   // 現在の表示枠幅 w, 高さ h
{ h = (h == 0) ? 1 : h;      // h が 0 ならば 1 にする
   glViewport(0, 0, w, h);   // 全表示範囲を指定
   glMatrixMode(GL_PROJECTION);// 投影変換行列を設定
    glLoadIdentity( );       // 単位行列を投影変換行列にセット
   glOrtho( -w,w,-h,h, -1.,1.);// 正射影の変換行列を掛ける
   glMatrixMode(GL_MODELVIEW); // 形状用変換行列
}

void display(void)
{ glClear(GL_COLOR_BUFFER_BIT);
    if( move )rote += 0.1; if( rote>0. ) rote-=360.;
   glLoadIdentity( ); // モデル視点変換行列を初期化
    glPushMatrix( );  // 現在の変換行列を退避
       glTranslatef( xtrans, ytrans,0. ); // 並行移動
       glRotatef( rote,0.,0.,1. ); // rote  度の回転
       glScalef( 200.,200.,200. );// 200 倍拡大
       glColor3f( 0.,0.,1. ); // 以降青色で表示
      glCallList( SQUARE );//     四角形表示
   glPopMatrix(); // 退避した変換行列を戻す
   glutSwapBuffers( ); // 表示画面を切り替え
}
```

```
void special_input(int key, int x, int y)
{ switch(key){   // 上下左右キーで物体を移動
      case GLUT_KEY_LEFT: xtrans -= 5.; break;
      case GLUT_KEY_RIGHT:xtrans += 5.; break;
      case GLUT_KEY_UP:     ytrans += 5.; break;
      case GLUT_KEY_DOWN:   ytrans -= 5.; break;
      default: return;
   }
   glutPostRedisplay(); // 描画
}

void key_input(unsigned char key, int x, int y)
{ switch(key){
     case 27: case 10:   case 13: exit(0); // ESC 等終了
     case ' ': move = 1 - move; //  スペースキーで回転
           if(move) glutIdleFunc(display); //  常時描画
           else glutIdleFunc(NULL); break; //  描画停止
     default:     break;
   }
}

int main(int argc, char **argv)
{
   glutInitWindowSize (800, 600); //  ウィンドウ大きさ
   glutInit(&argc, argv);       // グラフィックス初期化
  // RGBA モードで、ダブルバッファ
   glutInitDisplayMode(
                  GLUT_RGBA | GLUT_DOUBLE);

   glutCreateWindow(argv[0]);// 描画ウィンドウ生成
   myinit( );
   glutDisplayFunc(display);   // 描画用関数を指定
   glutReshapeFunc(reshape); // 再描画時に呼ばれる関数
    glutIdleFunc(NULL);  // 常時呼ばれる関数を指定(無し)
   glutSpecialFunc(special_input);  //   矢印キー割込関数
    glutKeyboardFunc(key_input);//キー割込関数
    glutMainLoop( );                    // 処理待ち
    return 0;
}
```



実行結果

スペースキーで回転、矢印キーで平行移動する

課題名 trans2D　http://www.coins.tsukuba.ac.jp/~fukui/CG/ex3.htm を参考にして、座標変換を使ったオリジナルの動画を作成すること。glPushMatrix( ), glPopMatrix( )を使うこと。