OpenGL

- typedef unsigned int GLenum
- typedef unsigned int GLbitfield
- typedef int GLint
- typedef int GLsizei
- typedef unsigned int GLuint
- typedef float GLfloat
- typedef float GLclampf
- typedef double GLdouble
- typedef void GLvoid
- void glutInit(int *argcp, char **argv)
  - GLUTライブラリを初期化します.
  - 「argc」と「argv」はmain関数の引数，すなわちコマンドライン引数を渡します．これらの引数は，コマンドラインのオプション指定時に用いられます.
- void glutInitDisplayMode(unsigned int mode)
  - ディスプレイの表示モードを設定します.
  - 「glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE)」のように書くと，「RGBAカラーモデル」で「ダブルバッファ」を使うという指定になります.
- void glutInitWindowSize(int width, int height)
  - ウィンドウの初期サイズを設定します.
  - 「width」はウィンドウの幅，「height」はウィンドウの高さになります.
- void glutInitWindowPosition(int x, int y)
  - ウィンドウの左上の位置を指定する．引数は共にピクセル値.
- int glutCreateWindow(char *title)
  - ウィンドウを生成する．引数はそのウィンドウの名前となる.
- void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha)
  - 「glClear(GL_COLOR_BUFFER_BIT)」でウィンドウを塗りつぶす際の色を指定します.
  - 「red」「green」「blue」はそれぞれ「赤」「緑」「青色」の成分の強さを示すGLclampf型（float型と等価）の値で，0～1の間の値をもちます．1が最も明るく，この3つに(0,0,0)を指定すれば「黒色」になり，(1,1,1)を指定すれば「白色」になります.
  - 最後の「alpha」は「α値」と呼ばれ，OpenGLでは不透明度として扱われます（0で透明，1で不透明）．ここではとりあえず「1」にしておいてください.
- void glutMainLoop(void)
  - GLUTのイベントが発生するまで，待機状態になります.
- void glutSwapBuffers(void)
  - 描画の最後で記述する．この関数が実行されると，バックバッファの内容がフロントバッファに転送される.
- void glClear(GLbitfield mask)
  - 「mask」に指定したバッファのビットを初期化します.
  - 「glClear(GL_COLOR_BUFFER_BIT)」と指定すると「カラーバッファ」が初期化されます.

OpenGL

- void glutDisplayFunc(void (*)(void))
  - 引数は開いたウィンドウ内に描画する関数へのポインタです．ウィンドウが開かれたり，他のウィンドウによって隠されたウィンドウが再び現われたりしてウィンドウを再描画する必要があるときに，この関数が実行されます．したがって，この関数内で図形表示を行います．
- void glutReshapeFunc(void (*)(int width, int height))
  - 引数には，ウィンドウがリサイズされたときに実行する関数のポインタを与えます．この関数の引数にはリサイズ後のウィンドウの幅と高さが渡されます．
- void glutKeyboardFunc(void (*)(unsigned char key, int x, int y))
  - 引数には，キーがタイプされたときに実行する関数のポインタを与えます．この関数の引数「key」には，タイプされたキーのASCIIコードが渡されます．また，「x」と「y」にはキーがタイプされたときのマウスの位置が渡されます．
- void glutMouseFunc(void (*)(int button, int state, int x, int y))
  - 引数には，マウス・ボタンが押されたときに実行する関数のポインタを与えます．この関数の引数「button」には，押されたボタン（GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON）が渡されます．引数「state」には，「押した」（GLUT_DOWN）のか「離した」（GLUT_UP）のかが渡されます．また，引数「x」と「y」にはその位置が渡されます．
- void glutPostRedisplay(void)
  - ウィンドウを再描画します．glutDisplayFunc()で登録したコールバック関数が呼び出されます．
- void glViewport(GLint x, GLint y, GLsizei width, GLsizei height)
  - 「ビューポート」を設定します．「ビューポート」とは，開いたウィンドウの中で，実際に描画される領域のことをいいます．正規化デバイス座標系の2点(-1,-1)，(1,1)を結ぶ線分を対角線とする矩形領域がここに表示されます．最初の2つのGLint型（int型と等価）の引数「x,y」にはその領域の左下隅の位置，後の2つのGLsizei型（int型と等価）の「width」と「height」には，それぞれ幅と高さをデバイス座標系の値，すなわちディスプレイ上の画素数で指定します．glutReshapeFuncで指定されたコールバック関数の引数「width,height」にはそれぞれウィンドウの幅と高さが入っていますから，glViewport(0,0,width,height)はリサイズ後のウィンドウの全面を表示領域に使うことになります．

OpenGL

- void glBegin(GLenum mode)
  void glEnd(void)
  - 「glBegin」と「glEnd」の間に指定した頂点座標を使って，描画を行います．
  - 描画内容は「mode」に指定します．「mode」には「GL_LINE_STRIP」などが指定できます．
- void glVertex2d(GLdouble x, GLdouble y)
  - 2次元の座標値を設定します．
  - 引数はGLdouble型の(x, y) で指定します．
- void glColor3d(GLdouble red, GLdouble green, GLdouble blue)
  - これから描画するものの色を指定します．
  - 引数の型はGLdouble型で，「red」「green」「blue」にはそれぞれ「赤」「緑」「青」の強さを「0～1」の範囲で指定します．

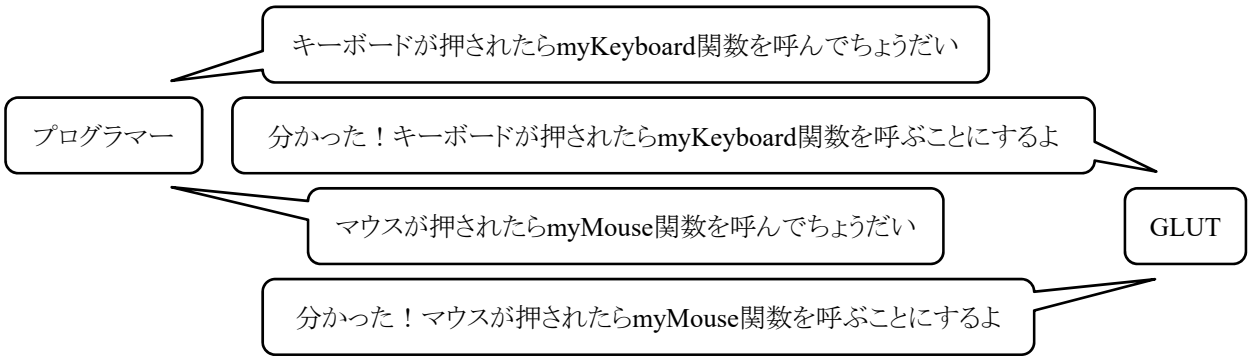
OpenGL

- void glRasterPos2d(GLdouble x, GLdouble y)
  - ラスタ位置を決定する．位置は，現在のモデルビュー行列あるいは投影行列を用いて，スクリーン座標系に射影される．
- void glutBitmapCharacter(void *font, int character)
  - fontで指定されたフォントを用いて，ASCIIコードで与えられた文字characterを描く．fontにはGLUT_BITMAP_TIMES_ROMAN_24などがある．

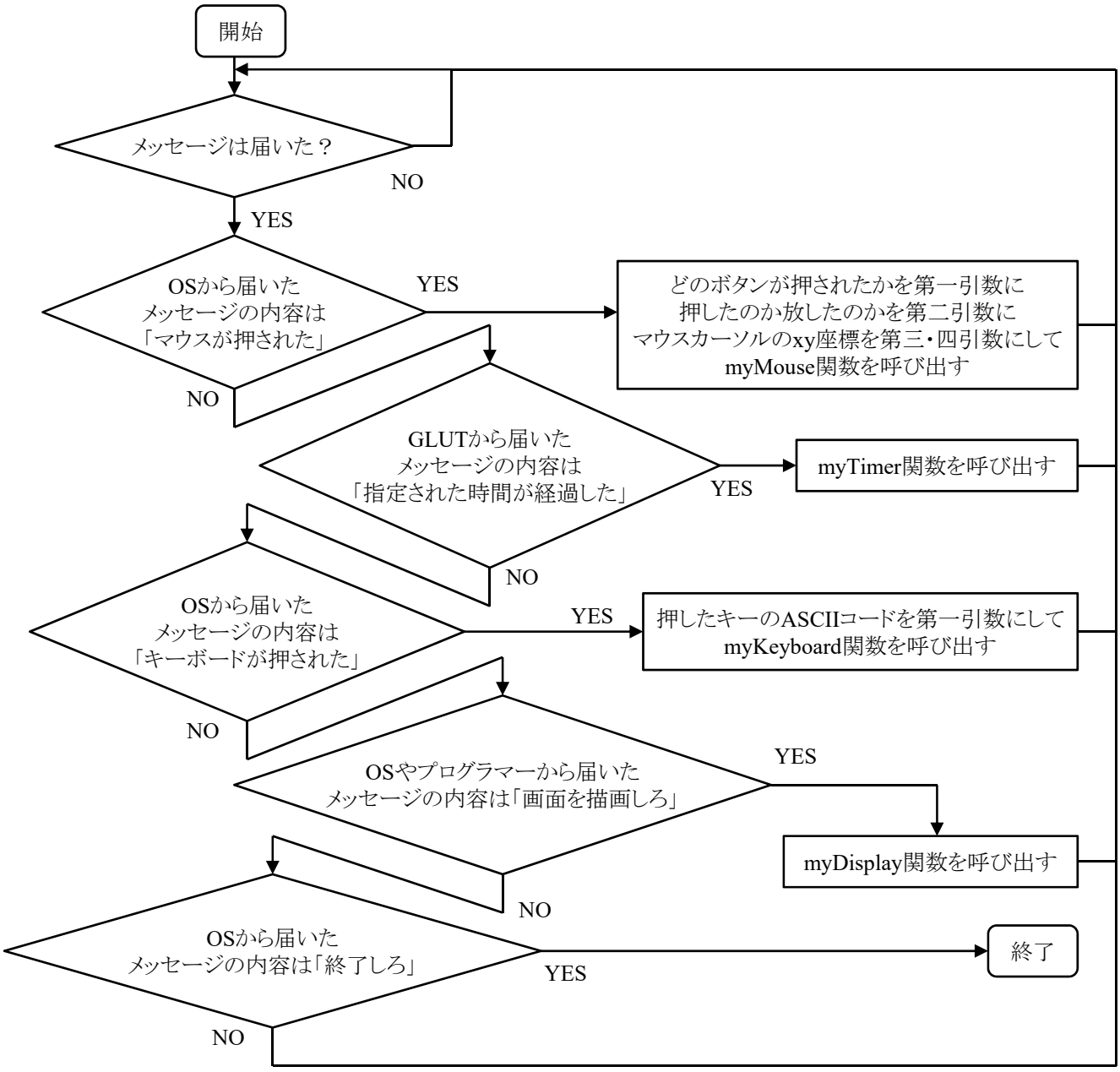
イベントドリブン型プログラミング

- glutMainLoopは無限ループしているだけで，メッセージが届くのをひたすら待ち続けます
- メッセージを受け取ったら，メッセージに応じた処理をして，また再び無限に待機します

コールバック関数の指定



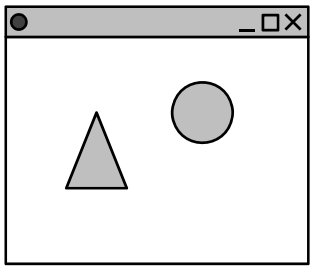
ループ中の動作の例



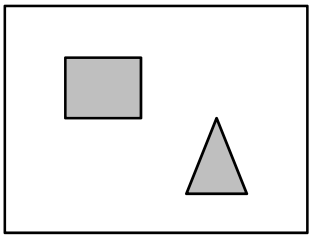
画面の描画

- glClearで画面を消去してglutSwapBuffersで描画します

描画の流れ

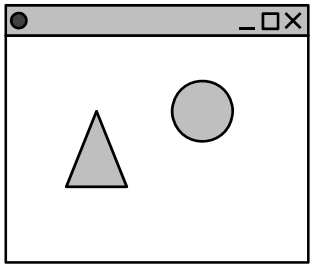


ウィンドウ表示用のバッファ



メモリ内にあるバッファ

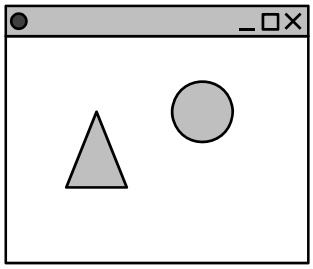
glClear(GL_COLOR_BUFFER_BIT);
画面消去



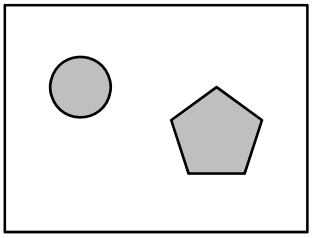
ウィンドウ表示用のバッファ

メモリ内にあるバッファ

glBegin～glEnd
図形の描画

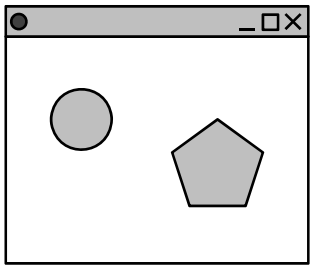


ウィンドウ表示用のバッファ

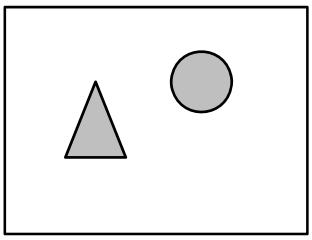


メモリ内にあるバッファ

glutSwapBuffers();
ウィンドウに表示



ウィンドウ表示用のバッファ



メモリ内にあるバッファ

```
void ディスプレイコールバック関数()
{
 // 画面消去
 glClear(GL_COLOR_BUFFER_BIT);

 // 図形の描画
 glBegin(GL_QUADS);
 glVertex3d(-1.0, -1.0, 0.0);
 glVertex3d(1.0, -1.0, 0.0);
 glVertex3d(1.0, 1.0, 0.0);
 glVertex3d(-1.0, 1.0, 0.0);
 glEnd();

 // ウィンドウに表示
 glutSwapBuffers();
}
```

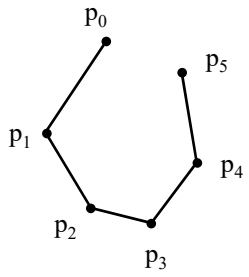
点，線，ポリゴンの描画

- 点，線，ポリゴンを描くのに，次のようにglBegin()とglEnd()および，glVertex*()を用いる．
- glBegin(mode);
  glVertex*(p0);
  glVertex*(p1);
  .....
  glVertex*(pn);
  glEnd();
- modeは描く図形の種類を指定し，p0，p1，…，pnは座標位置を意味する．

modeの種類

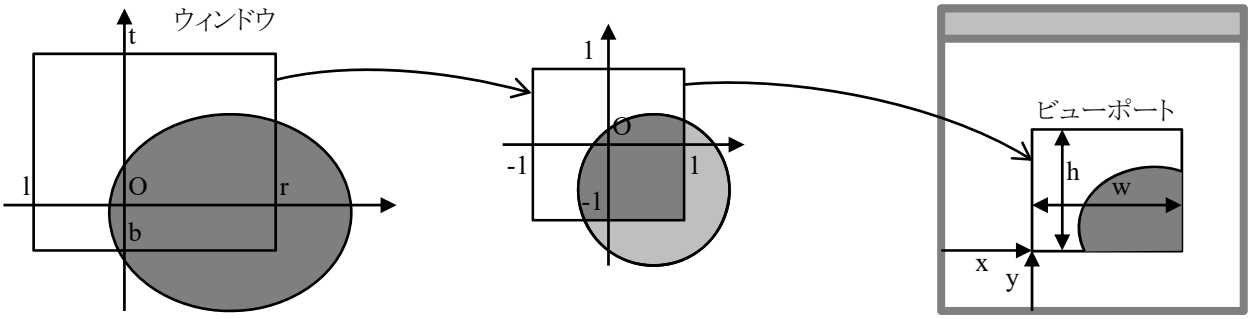
- GL_LINE_STRIP　　最初の頂点から最後の頂点まで線分を連結して描画する．

ポリゴン描写の注意点

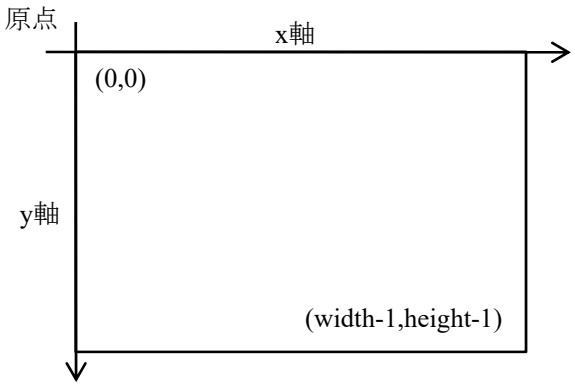


GL_LINE_STRIP

ウィンドウとビューポート



ワールド座標系　　正規化デバイス座標系　　デバイス座標系(ウィンドウ)
(左下が原点)



マウスの座標系

マウス

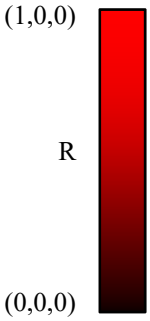
- マウスのボタンを押したか離したか，そのときの画素位置はglutMouseFuncで指定するコールバック関数で確かめる

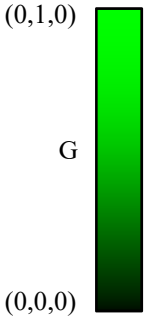
コールバック関数の設定
```
glutMouseFunc(myMouse);
```

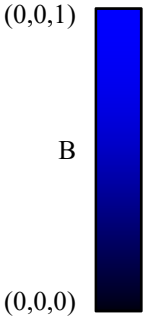
```
void myMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        printf("左ボタンが押された位置は(%d,%d)¥n", x, y);
    }
    if (button == GLUT_LEFT_BUTTON && state == GLUT_UP) {
        printf("左ボタンが放された位置は(%d,%d)¥n", x, y);
    }
}
```

色の表現

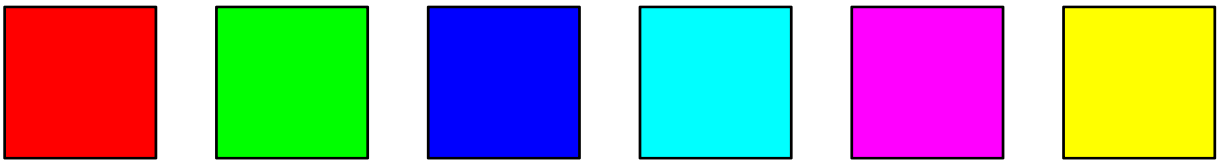
- 光の三原色で表す
- R: Red 赤, G: Green 緑, B: Blue 青
- glClearColor関数で背景色を指定，glColor3d関数で描画する図形の色を指定します
- OpenGLのこれらの関数の引数は，浮動小数点の0～1の値で表します
  - ちなみに，画面や画像ファイルなどの画素は通常，RGBそれぞれ8bitずつ，合計24bitで表します．整数で0～255の値で表します．OpenGLは0～1ですのでご注意ください．
- 0が暗くて，1が明るいです







(R,G,B)=(0,0,0)　　(R,G,B)=(0.5,0.5,0.5)　　(R,G,B)=(1,1,1)



(1,0,0)　(0,1,0)　(0,0,1)　(0,1,1)　(1,0,1)　(1,1,0)

(1,0,0)　(1,0.5,0.5)　(1,1,1)　　(0,1,0)　(0.5,1,0.5)　(1,1,1)　　(0,0,1)　(0.5,0.5,1)　(1,1,1)

(1,0,0)　(1,1,0)　(0,1,0)　(0,1,1)　(0,0,1)　(1,0,1)　(1,0,0)

図形の色

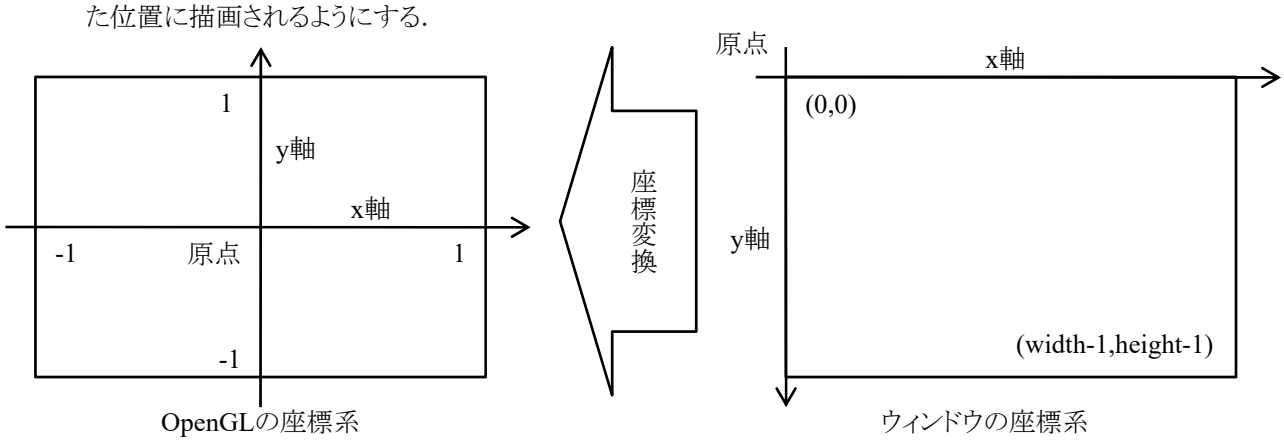
glColor3d関数（ライティングしない場合）

- 図形の色はglColor3d関数で指定します.

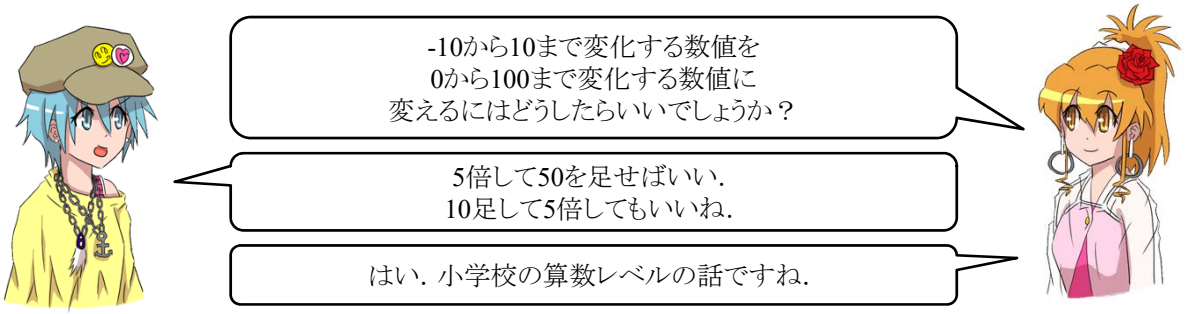
例2
```
glColor3d(1.0, 0.0, 0.0);
glBegin(GL_QUADS);
glVertex3d(-1.0, -1.0, 0.0);
glVertex3d(1.0, -1.0, 0.0);
glVertex3d(1.0, 1.0, 0.0);
glVertex3d(-1.0, 1.0, 0.0);
glEnd();
```

お絵かきソフト 兼 タイピング練習ソフト

- キーボードを押すとその文字が表示される．マウスをクリックすると線画が描ける．表示する文字の数と表示する線分の数にはリミットをもうける．マウスによる描画では，ウィンドウのサイズを変えても，クリックした位置に描画されるようにする．



- マウスでクリックしたときの(x,y)座標をOpenGLの座標系に変換せよ．クリックした位置に線分の端点が表示されるようにせよ．
  - glutMouseFuncにmyMouseFuncという関数を指定している．つまり，マウスを押すか放すかするとmyMouseFunc関数が呼び出される．引数のbuttonがGLUT_LEFT_BUTTONで，stateがGLUT_DOWNだった場合，マウスの左ボタンが押されたことを意味する．このとき，引数のxとyには，左ボタンをクリックされたときの座標が入っている．上図で言うと，右の座標系での座標である．
  - 線分を描画する処理はmyDisplay関数に記述されている．glVertex2dの引数として，配列xListとyListに格納された座標が使われている．この座標は上図で言うと，左の座標系での座標である．
  - 上図の右に書かれている「width」「height」はウィンドウのサイズを表し，プログラム中では「winw」「winh」という変数に格納されている



- 「目的が分からない」という質問が多いのでもう一度説明します．右上の「ウィンドウの座標系」から左上の「OpenGLの座標系」に変換するのが目的です．
- プログラムの変数との対応関係を言うと，右上の「ウィンドウの座標系」のxy座標がxとy，左上の「OpenGLの座標系」のxy座標がxList[pointnum]とyList[pointnum]です．つまり，x,y,winw,winh,+,-,*,/,数字,(,),doubleなどを使って座標を適切に計算してxList[pointnum]にx座標を，yList[pointnum]にy座標を代入することが目的です．
- それでも「目的が分からない」という質問が多いので分かりやすく目的を伝えます．「お絵かきソフトを作ってください」というのが目的です．

int型同士の割り算は小数点以下が省略されるよ！
実数同士の割り算をするときは必ず型キャストをしよう！
変数の前に(double)を付けるとちゃんと実数として計算されるよ！

プロトタイプ

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

int winw, winh;
const int TEXTMAX = 40;
const int POINTMAX = 10;
char text[TEXTMAX];
double xList[POINTMAX];
double yList[POINTMAX];
int textnum;
int pointnum;

void myDisplay()
{
    int i, j;

    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3d(0.0, 0.0, 0.0);
    glBegin(GL_LINE_STRIP);
    for (i = 0; i < POINTMAX; i++) {
        j = (pointnum + i) % POINTMAX;
        if (xList[j] >= -1.0 && yList[j] >= -1.0) {
            glVertex2d(xList[j], yList[j]);
        }
    }
    glEnd();
    glRasterPos2d(-0.9, -0.7);
    for (i = 0; i < TEXTMAX; i++) {
        j = (textnum + i) % TEXTMAX;
        if (text[j] != '¥0') {
            glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, text[j]);
        }
    }
    glutSwapBuffers();
}

void myMouseFunc(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        xList[pointnum] = -10.0 /* [課題] 適切にx座標を計算せよ */;
        yList[pointnum] = -10.0 /* [課題] 適切にy座標を計算せよ */;
        pointnum = (pointnum + 1) % POINTMAX;
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
    text[textnum] = key;
    textnum = (textnum + 1) % TEXTMAX;
    glutPostRedisplay();
}
```

PDFファイルのテキストをそのままコピペするときは
PDFファイルではバックスラッシュと円マークの文字コードが違うので円マークはちゃんと自分でキーボードから打つこと！

プロトタイプ

```
void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    int i;
    winw = 640;
    winh = 480;
    pointnum = 0;
    for (i = 0; i < POINTMAX; i++) {
        xList[i] = -10.0;
        yList[i] = -10.0;
    }
    textnum = 0;
    for (i = 0; i < TEXTMAX; i++) {
        text[i] = '¥0';
    }
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(winw, winh);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutMouseFunc(myMouseFunc);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

PDFファイルのテキストをそのままコピペするときは
PDFファイルではバックスラッシュと円マークの文字コードが違うので円マークはちゃんと自分でキーボードから打つこと！