平成 19 年度 機械情報工学科演習
コンピュータグラフィクス (3)
3DCG とインタラクション

担当：谷川 智洋 講師，西村邦裕 助教
TA：仲野 潤一，山口 真弘，藤澤 順也

2007 年 12 月 14 日

1 演習の目的

本日の演習では，前々回の OpenGL の基礎，前回の OpenGL と GLUI の組み合わせ，テクスチャマッピングを元にし，三次元コンピュータグラフィックスとのインタラクションについて学ぶ．一つは，EyeToy を用いて，動画を三次元コンピュータグラフィックスに貼り込む手法を学習する．もう一つは，Wii リモコンを用いて，三次元グラフィックスとのインタラクション手法，すなわち，ボタンを押すことや腕ふりなどの動作と OpenGL 内での動作との間の関係，について学習する．

1.1 資料など

本日の演習の資料などは，

```
http://www.cyber.t.u-tokyo.ac.jp/~kuni/enshu2007/
```

においてある．

本日使用するソースファイルも同じ場所からダウンロードすること．前々回，前回の資料を参照したい場合も同じ場所を参照のこと．参考となる URL もリンクが貼ってあるため，参照のこと．ネットワークにつながらない者は，教員に USB メモリを借りてデータをコピーすること．

1.2 出席・課題の確認について

出席の確認は，課題の確認によって行う．課題が終了したら，教員・TA を呼び，指示に従って実行して説明せよ．

2 動画の取り扱い

画像処理で学習した EyeToy を用いた動画の取り扱いを一歩進んで，OpenGL に動画を貼ることについて説明する．

Linux において，動画を利用するためには，Video4Linux と呼ばれるビデオキャプチャデバイスを使うための API 仕様を利用する．Video4Linux を用いることにより，USB カメラ，IEEE1394 カメラなどから画像を読み込むことができるとともに，ビデオキャプチャボードからの読み込みもできる．また，チューナー操作のための API 仕様などもそろっている．

Linux のカーネルに Video4Linux のドライバがインストールされており，かつ，ビデオキャプチャ用のドライバ（モジュール）が正しくロードされていた場合，EyeToy などのカメラを接続した状態で，xawtv 等のソフトを用いて，動作を確認することができる．

2.1 動画を用いたプログラミング

動画の読み込み

Video4Linux のドライバとして，ARToolKit などを開発した加藤博一先生（奈良先端科学技術大学院大学）や中澤篤志先生（大阪大学）のドライバをベースにして，説明をする．参照: sample-texture.c

```
#include "video.h"

#define VIDEO_WIDTH     640
#define VIDEO_HEIGHT    480

/* カメラから画像を取得する初期設定 */
int myInitFunc(void )
{
  /* メモリ確保 */
  image = (IMAGE *)mallocImage(VIDEO_WIDTH, VIDEO_HEIGHT, 3);

  /* ビデオデバイスのオープン */
  if( arVideoOpen("-dev=/dev/video0 -width=640 -height=480") != 0 )
    {
      printf("cannot open video device\n");
      return 0;
    }

    ・・・
```

```
    /* ビデオキャプチャのスタート */
    arVideoCapStart();

}



/* アイドル時の入力のコールバック関数 */
void idleFunc( void )
{
  unsigned char *ptr, tmp;

  /* ビデオキャプチャを行う．画像は ptr に格納 */
  ptr = arVideoGetImage();
  if( ptr == NULL )
      return;
  /* ptr に格納された画像を image にメモリコピー */
  // store grabimage to mainmemory.
  memcpy( image->data, ptr, image->width*image->height*3 );

  /* 次のフレームのキャプチャを行う */
  // capture next frame.
  arVideoCapNext();

#if 1
  /* RGB の格納順番を変更する */
  /* change BGR -> RGB bitplane */
  ptr = image->data;
  for( i = 0; i < image->width*image->height; i++ )
    {
      tmp  = *ptr;
      *ptr = *(ptr+2);
      *(ptr+2) = tmp;

      ptr+= 3;
    }
#endif

}
```

上記の方法では，arVideoOpen を用いて，ビデオデバイスをオープンする．引数として，
-dev: オープンするデバイスのファイル名 (/dev/video0 など) を指定
-width, -height: 画像サイズを指定
-channel: チャネル（1:コンポジット端子，2:SVideo など選択可能）
-mode: モード (ビデオ信号として NTSC,PAL,SECAM を選択可能)
を指定する．

```
arVideoOpen("-dev=/dev/video0 -width=640 -height=480")
```

さらに，arVideoCapStart でビデオキャプチャをスタートし，arVideoGetImage でビデオキャプチャの実行，arVideoCapNext で次のフレームのビデオキャプチャ，arVideoCapStop でビデオキャプチャの終了である．

```
arVideoCapStart();
arVideoGetImage();
arVideoCapNext();
arVideoCapStop();
```

キャプチャをしたデータを表示すると，下記のように Window 内に表示することが可能となる（図 1).

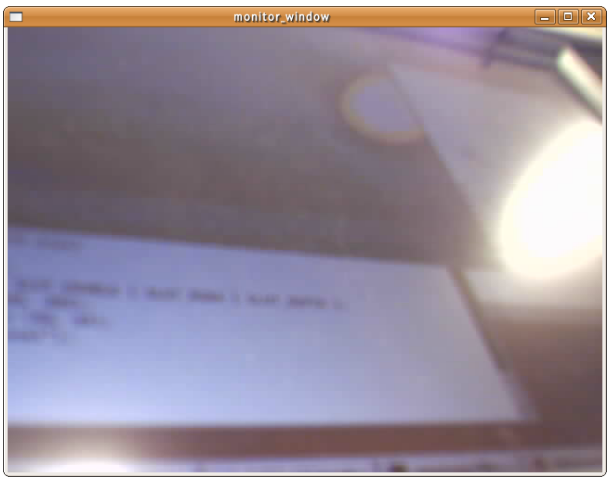

図 1 動画読み込みと描画

読み込んだ動画の描画

前回，学習した glDrawPixels() を呼び出して，キャプチャした画像を読み込むことができる．

```
void glDrawPixels(GLsizei width, GLsizei height, GLenum format,
                  GLenum type, GLvoid *pixels);
void glRasterPos{234}{sifd}(TYPE x, TYPE y, TYPE z, TYPE w);
```

また，読み込んだ画像データをテクスチャとしてマッピングすることで，動画を貼ることが可能である．テクスチャマッピングは通常の方法と同じである．参照: sample-texture.c

```
/* 描画部分について */
void displayFunc( void )
{
    glClear( GL_COLOR_BUFFER_BIT );

    /* glDrawPixels を用いる場合はコメントアウト */
    /*
      glRasterPos2i( 0, 0 );
      glDrawPixels( image->width, image->height, image->format,
      GL_UNSIGNED_BYTE, image->data );
    */

    /* テクスチャマッピングを用いる場合 */
    glEnable( GL_TEXTURE_2D );
    glColor3f( 1.0, 1.0, 1.0 );
    glBindTexture( GL_TEXTURE_2D, tex_index );

    glPushMatrix();
    glBegin( GL_TRIANGLE_STRIP );
    glTexCoord2f(1.0, 1.0);
    glVertex3f(tex_width*0.75, 0.0, 0.0);
    glTexCoord2f(0.0, 1.0);
    glVertex3f(tex_width*0.25, 0.0, 0.0);
    glTexCoord2f(1.0, 0.0);
    glVertex3f(tex_width*0.75, tex_height*0.5, 0.0);
    glTexCoord2f(0.0, 0.0);
    glVertex3f(tex_width*0.25, tex_height*0.5, 0.0);
    glEnd();
```

```
    glPopMatrix();
    glDisable( GL_TEXTURE_2D );

    glFlush();
    glutSwapBuffers();
}


/* アイドル時の入力のコールバック関数 */
void idleFunc( void )
{
  unsigned char *ptr;

  /* ビデオキャプチャを行う．画像は ptr に格納 */
  ptr = arVideoGetImage();
  if( ptr == NULL )
      return;

  /* ptr に格納された画像を image にメモリコピー */
  // store grabimage to mainmemory.
  memcpy( image->data, ptr, image->width*image->height*3 );

  /* 次のフレームのキャプチャを行う */
  // capture next frame.
  arVideoCapNext();

  /* テクスチャの設定 */
  glPixelStorei( GL_UNPACK_ALIGNMENT, 1 );
  glBindTexture( GL_TEXTURE_2D, tex_index );
  glTexSubImage2D( GL_TEXTURE_2D, 0, 0, 0,
                   image->width, image->height,
                   image->format, GL_UNSIGNED_BYTE, image->data );

   glutPostRedisplay();
}
```

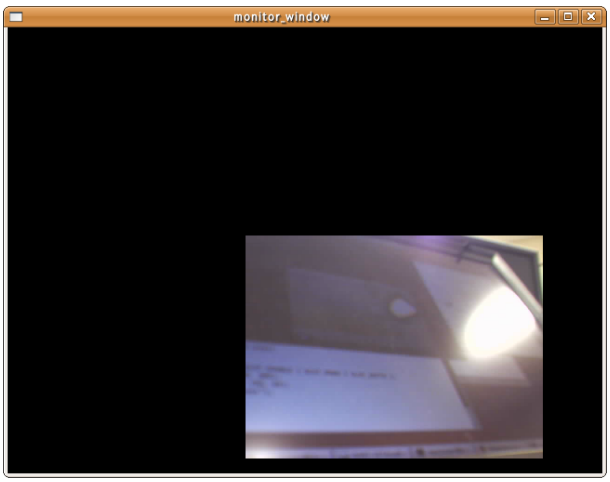

図 2　動画：2 次元へのテクスチャマッピング

テクスチャマッピングを 3 次元コンピュータグラフィックス上に行うことで，動画を 2 次元 (図 2 だけではなく 3 次元にもマッピング (図 3) することができる．カメラを接続しているので，リアルタイムに動画が変化する．

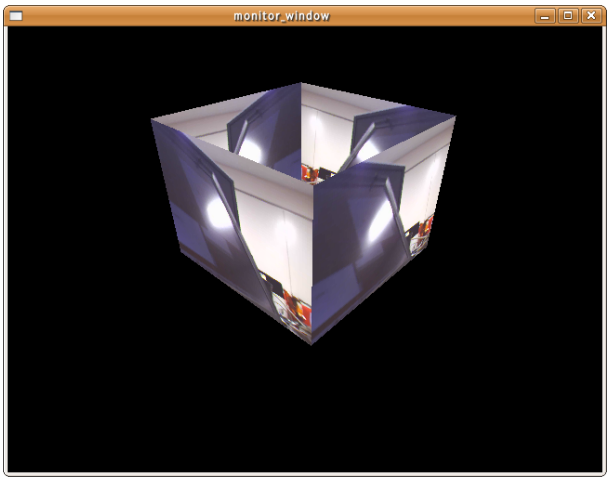

図 3　動画：3 次元の箱へのテクスチャマッピング

◇ **課題** 1 ◇

1. video-sample.tar.gz をダウンロードし，コンパイルせよ．EyeToy を接続した後，実行して，画像が表示されるか確認せよ．また，Window サイズが変わってもきちんと表示されるかを確認せよ．EyeToy の動作確認の方法としては,「アプリケーション」→「サウンドとビデオ」→「XawTV」を起動で可能である．
2. video-texture.tar.gz をダウンロードし，実行せよ．また，テクスチャのサイズを変更したり，テクスチャの形を変形してみよ．
3. video-cube.tar.gz をダウンロードし，実行せよ．図 3 に示すように，正面のみならず，右面・左面・裏面にもテクスチャを貼り，箱として見えるように変更せよ．
4. GLUI を用いて，UI を付けよ．（オプション）

3　三次元コンピュータグラフィックスとインタラクション

三次元コンピュータグラフィックスを操作するには，キーボード，マウスその他のインタフェースを用いる．本演習では，身体の動きも取得することができる Wii リモコンを用いたインタラクション手法の基礎について学習する．任天堂 Wii は家庭用ゲーム機であり，体を動かしたり，腕を振ったり，という身体的動作まで含めたインタフェースを提供しており，従来型のボタンや矢印キーなども含まれている．また，特徴としては，Bluetooth を用いた無線での操作が可能であり，Wii リモコンにケーブルがついておらず，操作が容易となっている点も特徴である．

3.1　Wii リモコン

Wii のインタフェースとしては，従来型のクラシックコントローラ，Wii リモコン，Wii リモコンに接続するヌンチャク，Wii Fit などが発売されている．その中でも Wii リモコンは，簡単な操作，直感的な操作を目指して開発されており，誰でも容易に使えるインタフェースの一つである．

Wii リモコンには，3 軸加速度センサ，スピーカ，スピーカ用アンプ，電源コントロール IC，サウンドデコーダ，Bluetooth 用 IC，振動モータ，ポインタ用 CMOS カメラ，似顔絵データ用メモリが搭載されている．特に，Bluetooth が搭載されていることにより無線での利用が可能であり，3 軸リニア加速度センサを搭載していることによって，腕を振る動作や傾ける動作などを検出することが可能である．

図 4　Wii リモコン（http://www.nintendo.co.jp/wii/features/wii_remote.html）

3.2　CWiid

CWiid（http://abstrakraft.org/cwiid/）は，Linux 用に C で書かれた Wii リモコン（Wiimote) 用のツールであり，イベントの API などが含まれている．Wii リモコンの API や Wiimote ライブラリ，GTK をベースにした GUI インタフェースなどが含まれている．詳細は，下記のフォルダに入っているので参考にすること．

```
/home/mechuser/ubuntu/manual_install/cwiid-0.5.03/
```

今回の演習では，センサーバーなどないため，CMOS カメラなどは用いず，またヌンチャクなども用いずに行う．また，スピーカ部分も用いない．ボタンや加速度，振動を用いることとする．

コマンドの送信

Wii リモコンへのコマンドを送信するには，wiimote_command を用いる．

state が 1 で ON，0 で OFF になる．

command では， WIIMOTE_CMD_RUMBLE が振動，WIIMOTE_CMD_LED が LED である．

```
wiimote_command(wiimote, command, int state);
```

設定としては，下記の部分を書く．

```
void set_report_mode(void)
{
  uint8_t rpt_mode;

  rpt_mode = WIIMOTE_RPT_STATUS | WIIMOTE_RPT_BTN;

  rpt_mode |= WIIMOTE_RPT_ACC;
  wiimote_command(wiimote, WIIMOTE_CMD_RPT_MODE, rpt_mode);
}
```

Wii リモコンからのコールバック

Wii リモコンからのコールバックとしては，$unionwiimote\_mesgmesg\_array$ に入っている．$mesg\_array\text{-}type$ を用いて加速度とボタンの情報を識別する．acc_zero, acc_one は $zeropoint$ と $1Gpoint$ の初期値である
.

```
void wiimote_callback(int id, int mesg_count,
                      union wiimote_mesg *mesg_array[])
{
  int i;

  for (i=0; i < mesg_count; i++) {
    switch (mesg_array[i]->type) {
    case WIIMOTE_MESG_ACC:
      wiimote_acc(&mesg_array[i]->acc_mesg);
      break;
    case WIIMOTE_MESG_BTN:
      wiimote_btn(&mesg_array[i]->btn_mesg);
      break;

    default:
      break;
    }
  }
}
```

加速度とボタン

Wii リモコンからの値としては，$mesg->x, mesg->y, mesg->z$ に格納されているが，初期値との差をとることによって，加速度を計算することができる．

```
void wiimote_acc(struct wiimote_acc_mesg *mesg)
{
  printf("x: %d, y: %d, z: %d\t", mesg->x, mesg->y, mesg->z);
  double a_x, a_y, a_z;

  a_x = ((double)mesg->x - acc_zero.x) /
    (acc_one.x - acc_zero.x);
  a_y = ((double)mesg->y - acc_zero.y) /
    (acc_one.y - acc_zero.y);
  a_z = ((double)mesg->z - acc_zero.z) /
    (acc_one.z - acc_zero.z);
}
```

Wii リモコンからのボタンの値は，下記の手法で取得できる．

```
void wiimote_btn(struct wiimote_btn_mesg *mesg){

  if(mesg->buttons & WIIMOTE_BTN_UP){

  }
```

ボタンは，下記の引数で取得できる．
WIIMOTE_BTN_UP：十字キーの上ボタン
WIIMOTE_BTN_DOWN：十字キーの下ボタン
WIIMOTE_BTN_RIGHT：十字キーの右ボタン
WIIMOTE_BTN_LEFT：十字キーの左ボタン
WIIMOTE_BTN_PLUS：＋ボタン
WIIMOTE_BTN_MINUS：－ボタン
WIIMOTE_BTN_A：A ボタン
WIIMOTE_BTN_B：B ボタン
WIIMOTE_BTN_HOME：HOME ボタン
WIIMOTE_BTN_1：1 ボタン
WIIMOTE_BTN_2：2 ボタン

Wii リモコンの軸

Wii リモコンの軸は下記の通りである．

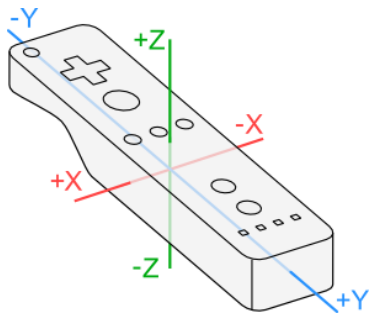

図 5 Wii リモコンの軸（http://wiibrew.org/images/9/9e/Wiimote_axis2.png）

Yaw, Pitch, Roll

参考までに，Yaw, Pitch, Roll について図 6 に示す．クォータニオン（四元数）を用いることで簡単に計算できるようになるが，今回は省略する．

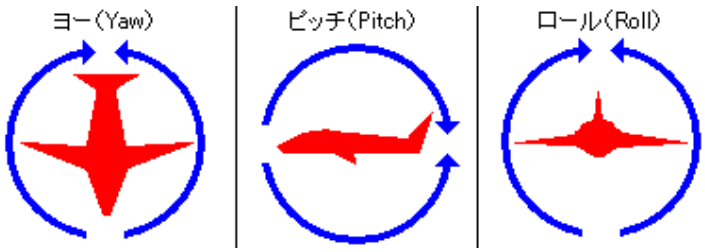

図 6 Yaw,Pitch,Roll（http://sorceryforce.com/programing/mdx/direct3d/stepup/quaternion.html）

CWiid を用いたサンプルプログラム

CWiid ライブラリを使用する Makefile の例は下記の通りである．cwiid-0.5.03/wiimote のフォルダを参照し，wiimote ライブラリを参照する．

```
CC  = g++

COMMON  = .
PACKAGE_VERSION = 0.5.03
WIIMOTE_DIR  = /home/mechuser/ubuntu/manual_install/cwiid-0.5.03/wiimote

RM  = rm -f
TARGET  = sample

OBJS  = sample.o

CFLAGS  = -Wall -I${WIIMOTE_DIR} \
#-DCWIID_VERSION=\"${PACKAGE_VERSION}\"
LDFLAGS = -L${WIIMOTE_DIR}
LDLIBS  = -lglut -lGL -lGLU -lwiimote



.c.o:
${CC} -c ${CFLAGS} $<

${TARGET}: ${OBJS}
${CC} -o ${TARGET} ${OBJS} ${LDLIBS} ${LDFLAGS}

clean:
${RM} ${TARGET} *.o *~
```

◇ **課題** 2 ◇

1. kadai.tar.gz をダウンロードせよ．kadai1 のフォルダに入っている sample.c をコンパイルし，実行せよ．実行方法は Bluetooth のアドレス 00:00:00:00:00:00 を引数にとり，実行する．Bluetooth のアドレスは貸し出した Wii リモコンに書かれている．例： ./sample 00:00:00:00:00:00
   (a) Wii リモコンを傾けてして，赤いボールが移動することを確認せよ．また，十字キー，＋ボタン，－ボタンで赤いボールが動くことを確認せよ．また，A ボタン，B ボタンなどでボールの色が変化することを確認せよ．
   (b) キーボードの「1」と「!」,「2」と「”」を押して，Wii リモコンの LED の 1 番，2 番が光る，消えることを確認せよ．また「r」と「R] を押して，Wii リモコンの振動が始まる・止まることを確認せよ．
   (c) ボタンを押すことによって，表示されている WireCube の回転がスタート，ストップできるようにせよ．
2. kadai2 のフォルダに入っている sample.c をコンパイルし，Wii リモコンを傾けることで，それに応じて Teapot が傾くことを確認せよ．
   (a) 十字キーを用いて大きさを変更できるようにせよ．
   (b) B ボタンを押すと，つかめて動かせるようにせよ．
   (c) 動きを移動平均などをとることにより安定化させよ（オプション）．
3. kadai3 のフォルダに入っている sample.c をコンパイルし，Wii リモコンを振ることによって球の動く方向が変化することを確認せよ.disp_flag によって，左向き，右向き，静止の状態が変化する．ホームボタンを押すと位置がリセットされる．
   (a) 加速度のパラメータを変化させ，手の動きに敏感あるいは鈍感に変更させてみよ．
   (b) 壁面を書いて，ぶつかったら反対向きに動くように変更せよ．壁をある位置に設定し，そこに来たら反対方向に動くようにさせる．これができると，壁にあたって帰ってきた球を，Wii リモコンを振ることによって打ち返せる感じになる．
   (c) ぶつかった際に振動が起きるようにせよ（オプション）．
   (d) 余裕があったら，壁あてゲームのように，球の速さが変わりながら，壁に当たって，戻ってきたら打ち返すようにせよ．（オプション）．

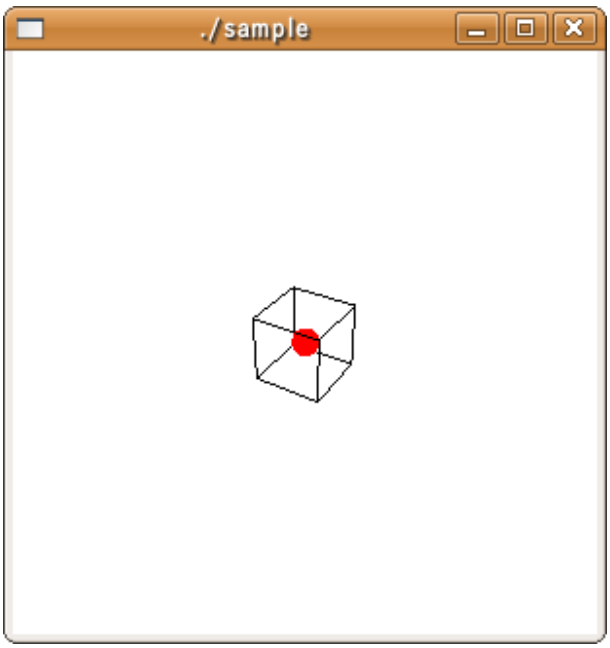

図 7　球が Wii リモコンの傾きにより動く例

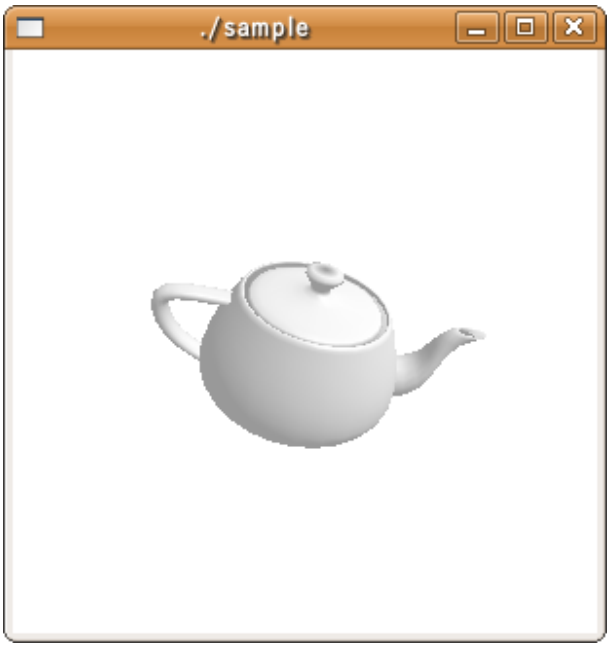

図 8　Teapot が Wii リモコンの向きにより動く例

付録 A　参考文献

伊藤邦朗，福田隆宏，”Wii リモコン”，日本機械学会誌，2007.12, Vol.110, No.1069, pp.6-7，2007.

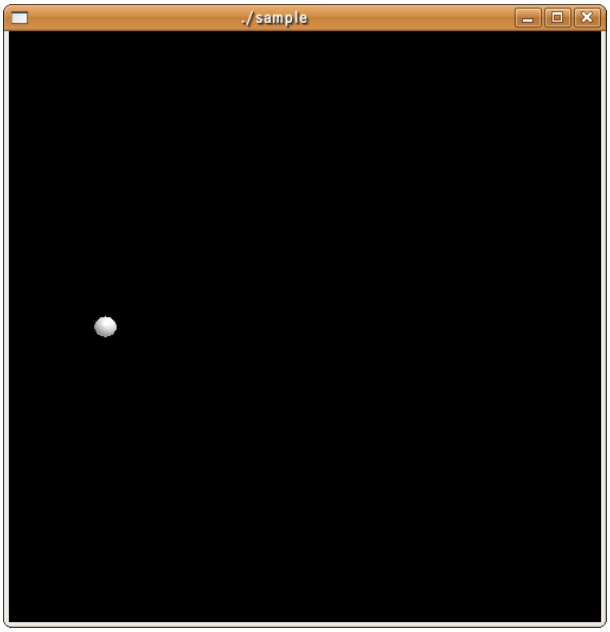

図 9　Wii リモコンを大きく振るとボールの動く向きが変更になる例