3次元コンピュータグラフィックスの基礎 -12

大野

CAVEのハードウェアシステム

- 県立大のCAVEのハードウェアシステム
  - 背面投影型スクリーン(正面スクリーン, 側面スクリーンx2)
  - 床面スクリーン
  - 液晶ステレオプロジェクター x4
  - 日本SGI社製ワークステーション(Xeon E5-2640v3 2.6GHz x2, メモリ:64GB, グラフィックス:Quadro K5200x4)
  - 液晶シャッター眼鏡(立体メガネ)
  - コントローラー(一般のシステムではワンド)
  - トラッキングシステム

トラッキング情報取得 -1

CAVELibが提供する関数により、立体メガネとワンド(県立大の場合コントローラー)の位置や向きを取得することができる。

トラッキング情報取得 -2

位置の取得のための関数
void CAVEGetPosition(CAVEID posid, float position[3])

posidには下記のものが入る。position[3]に結果が入ってくる（配列の引数は、関数で中身を変えられる）

CAVE_HEAD	実空間座標での頭(立体メガネ)の位置
CAVE_WAND	実空間座標でのワンド(コントローラー)の位置
CAVE_LEFT_EYE	実空間座標での左目の位置
CAVE_RIGHT_EYE	実空間座標での右目の位置
CAVE_HEAD_NAV	ナビゲーション座標での頭の位置
CAVE_WAND_NAV	ナビゲーション座標でのワンドの位置
CAVE_LEFT_EYE_NAV	ナビゲーション座標での左目の位置
CAVE_RIGHT_EYE_NAV	ナビゲーション座標での右目の位置

※ナビゲーション座標については後述

トラッキング情報取得 -2

方向の取得のための関数
void CAVEGetVector(CAVEID vectorid, float vector[3])

vectoridには下記のものが入る。vector[3]に結果が入ってくる

CAVE_HEAD_FRONT	CAVE_WAND_FRONT
CAVE_HEAD_BACK	CAVE_WAND_BACK
CAVE_HEAD_LEFT	CAVE_WAND_LEFT
CAVE_HEAD_RIGHT	CAVE_WAND_RIGHT
CAVE_HEAD_UP	CAVE_WAND_UP
CAVE_HEAD_DOWN	CAVE_WAND_DOWN
頭（立体メガネ）の実空間座標での方向（前方、後方、左、右、上、下）	ワンド（コントローラー）の実空間座標での方向（前方、後方、左、右、上、下）

トラッキング情報取得 -3

方向の取得のための関数
void CAVEGetVector(CAVEID vectorid, float vector[3])

ナビゲーション座標での方向を取得したい時はvectoridには下記のものを入れる

CAVE_HEAD_FRONT_NAV	CAVE_WAND_FRONT_NAV
CAVE_HEAD_BACK_NAV	CAVE_WAND_BACK_NAV
CAVE_HEAD_LEFT_NAV	CAVE_WAND_LEFT_NAV
CAVE_HEAD_RIGHT_NAV	CAVE_WAND_RIGHT_NAV
CAVE_HEAD_UP_NAV	CAVE_WAND_UP_NAV
CAVE_HEAD_DOWN_NAV	CAVE_WAND_DOWN_NAV
頭（立体メガネ）のナビゲーション座標での方向（前方、後方、左、右、上、下）	ワンド（コントローラー）のナビゲーション座標での方向（前方、後方、左、右、上、下）

※ナビゲーション座標については後述

トラッキング情報取得 -4

ワンドのボタン情報取得のための関数
int CAVEButtonChange(int button)

戻り値
0:　前回と比べて変化なし
1:　ボタンが押された
-1:　ボタンが離された

int button
1
2
3

トラッキング情報取得 -5

ワンドのボタン情報取得のためのマクロ

CAVEBUTTON1
CAVEBUTTON2
CAVEBUTTON3

値が
0:押されていない, 1:押されている

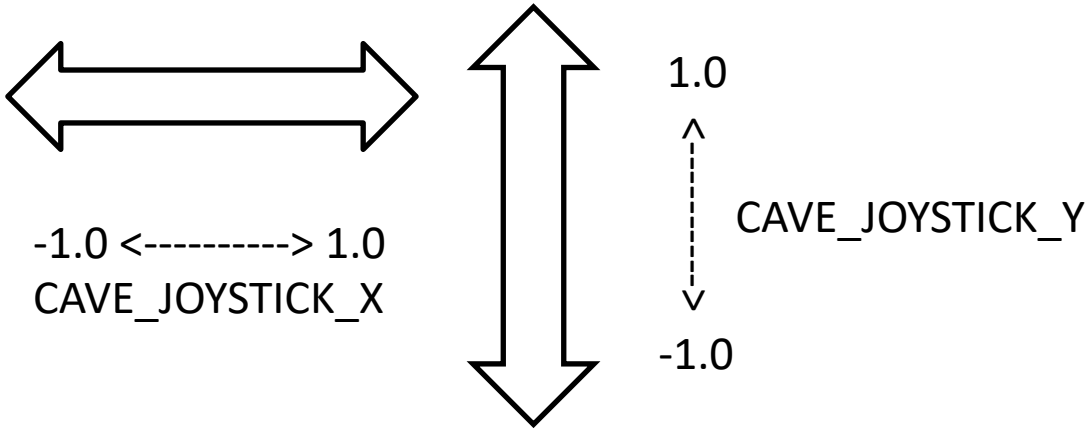
例
```
if(CAVEBUTTON1==1) {
  printf(“being  pressed¥n”);
 } else {
  printf(“not being pressed¥n”);
}
```

トラッキング情報取得 -6

ワンドのジョイスティックの傾き情報取得のためのマクロ

CAVE_JOYSTICK_X
CAVE_JOYSTICK_Y

値は、float 型
[-1.0, 1.0]

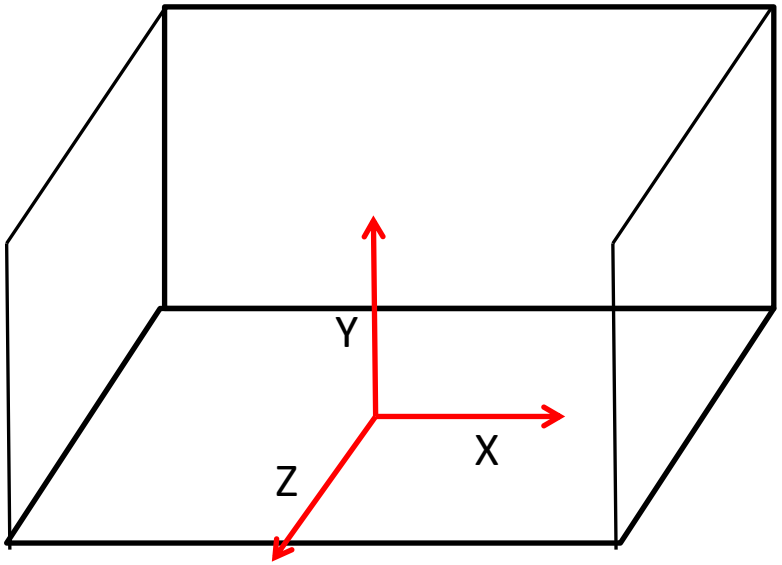


※ジョイスティックを倒してなくてもCAVE_JOYSTICK_X, _Y
には小さい値（ノイズ）が入っていることがあるので使用するときは、

```
if(fabs(CAVE_JOYSTICK_X)> 0.2){
  ....
}
```

などとするとよい。

ナビゲーション座標 -1

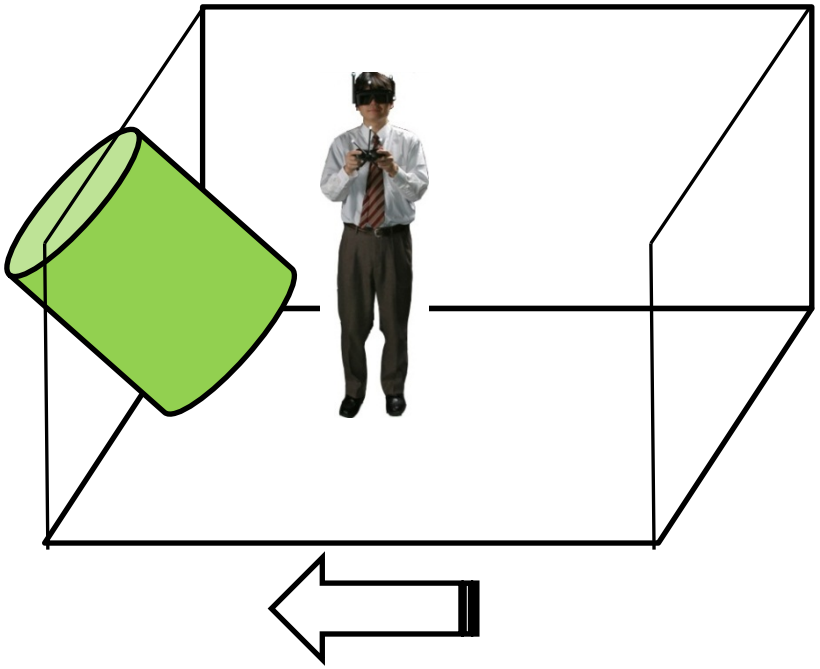


県立大のCAVEシステムのVR空間の大きさは、2m x 2m x 3.2mである。
VR空間内にビルなどを表示した場合、歩いて行って入れば、ビル内に入れる。
しかし、壁(や床)の向こうに表示されている場合は、屋内に入りようがない。

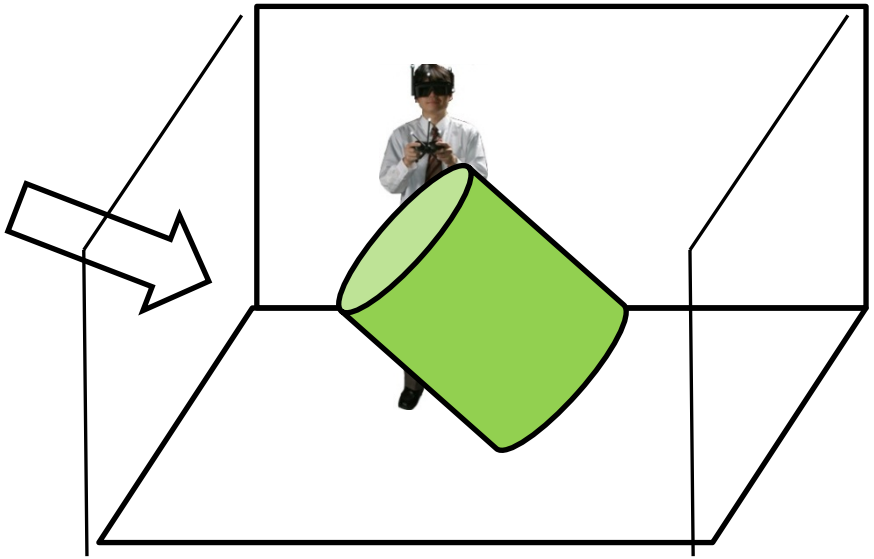
このような場合のために、ナビゲーション機能が用意されている。ナビゲーション機能とは、

ワンダ等を用いて、VR装置のグラフィックスに対する位置を移動させる機能

ナビゲーション機能 -2



ナビゲーションにより、VR装置を
CG空間の中で移動させる



円柱がVR装置の真ん中に来た（真ん中に表示された）

ナビゲーション機能 -3

ナビゲーションを実現する関数

- void CAVENavTranslate(float x, float y, float z)
  - 平行移動
  - Computation スレッドで実行
- void CAVENavRot(float angle, char axis)
  - 回転
  - angle (unit = degree)
  - axis: ‘x’, ‘y’ or ‘z’
  - Computation スレッドで実行
- void CAVENavTransform()
  - 実際の座標変換を行う。
  - Displayスレッドで実行

その他(CAVELibの関数)

- void CAVEUSleep(unsinged long t)
  - t micro seconds 停止
  - Computation のループに少し休憩させるために入れる
  - 注意 : void CAVESleep(float t): t seconds 停止
- boolean CAVEgetbutton(CAVE_DEVICE_ID device)
  - キーが押されたかどうか調べる。
  - deivce : CAVE_ESCKEY, CAVE_F1KEY～CAVE_F12KEY, CAVE_SPACEKEY, CAVE_AKEY～CAVE_ZKEYなど
  - boolean: true or false
- float CAVEGetTime()
  - CAVEがinitializeされてから何秒経過したか
  - アニメーションに重宝

その他(長さの単位)

- 描画などの単位について
  - デフォルトではFEETである

```
glBegin(GL_LINES);
 glVertex3f(0.0, 0.0, 0.0);
 glVertex3f(0.0, 1.0, 0.0);
glEnd();
```

↑　原点からy方向に、1フィートの長さの線をひく

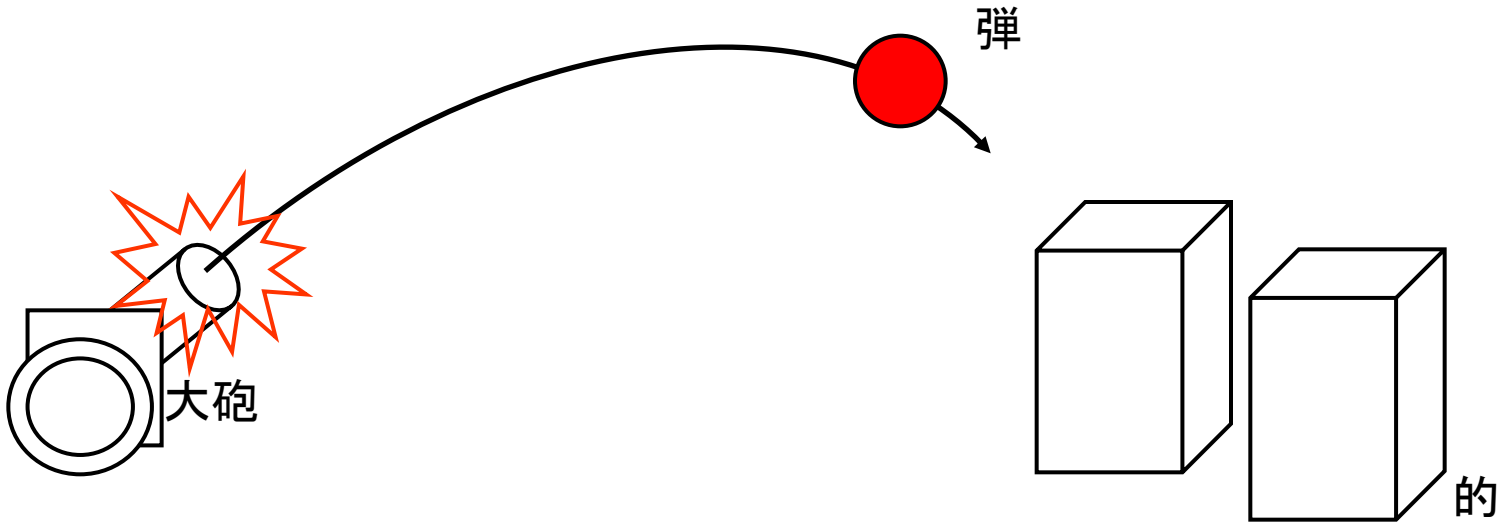
  - .cavercに、

    Units␣meters⏎

    の1行を入れると、単位がメーターになる

    使用可能な単位：feet, meters, inches, centimeters

ゲームを考えてみる(復習)



ルール

- ワンドを大砲に見立て、真ん中ボタンを押したら、ワンドの向いている方向に弾を発射する（初速度は与えておく）
- 軌道計算をして、的に当たったら的を消す
- 的が全部なくなるか、弾を撃ちつくしたらプログラム終了

ゲームを考えてみる(復習)

Computation スレッド

発射されていないとき

- ワンドのボタンが押されたかどうか調べ、ボタンが押されたなら、ワンドの位置と向きを調べ、弾を発射する(弾の状態を発射に変更)

発射されているとき

- 弾道計算
- 当たり判定
  - 当たった場合
  - 被弾した的を消す（的の数が0なら終了）
  - 弾の数を減らす（弾が0なら終了）
  - 弾の状態を「発射されていない」に戻す
- 地面に着弾
  - 弾の数を減らす（弾が0なら終了）
  - 弾の状態を「発射されていない」に戻す

Display スレッド

GLのInitで的と弾をディスプレイリストにする

的を表示

弾が発射されているとき

- 弾を表示

共有変数

- 弾の位置
- 弾の状態
  - 発射されているか否か
- 的の位置
- 的の存在状態

Displayスレッドが描画に専念できるようなプログラムを作る

CAVEプログラム演習

- Mayaへログインして、saber.c, saber2.c, snow.cをコンパイルしてみよう
- export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/CAVELIB/3.3/lib64をまず実行
- コンパイルには、makeコマンドを使う。saber2.c, snow.cのコンパイルはMakefileを修正して使う
- chmod␣764␣saber.sh⏎でsaber.shに実行権限をあたえ、./saber.shでsaberを実行する

1. saber.c, saber2.cでボタン、トラッキングデータの取得をマスターしよう
   1. ボタンを変えてみよう
   2. 光線の向きを変えてみよう
   3. ボタンを押すと目からレーザービームが出るようにしてみよう
2. snow.cで、ジョイスティック、ナビゲーションをマスターしよう
   1. ボタンを押している間、雪の動きが止まるようにしてみよう
   2. ジョイスティックをy方向に倒すと、視線方向に移動するようにしてみよう

共有メモリのゴミ -1

CAVELibのプログラムが異常終了した場合など、共有メモリやセマフォのゴミが残ってしまうことがある。

ゴミの有無は、**ipcs** コマンドで調べることができる。

残った共有メモリセグメントは

ipcrm␣–m␣(shmid)⏎

で消去できる

セマフォは、

ipcrm␣–s␣(semid)⏎

で消去できる

各自、自分が作ったゴミは消去すること

共有メモリのゴミ -2

```
#! /usr/bin/perl

$usr = $ENV{'USER'};
#print "user = $usr¥n";

@lines = `ipcs -m`;
foreach $line (@lines){
 $line =~ s/¥n//;
 @each = split(/[¥t¥s]+/, $line);
 if($each[2] eq $usr){
   $command = "ipcrm -m $each[1]";
   print "$command¥n";
   system($command);
 }
}
```

```
@lines = `ipcs -s`;
foreach $line (@lines){
 $line =~ s/¥n//;
 @each = split(/[¥t¥s]+/, $line);
 if($each[2] eq $usr){
   $command = "ipcrm -s $each[1]";
   print "$command¥n";
   system($command);
 }
}
```

共有メモリのゴミを削除するPerlスクリプト