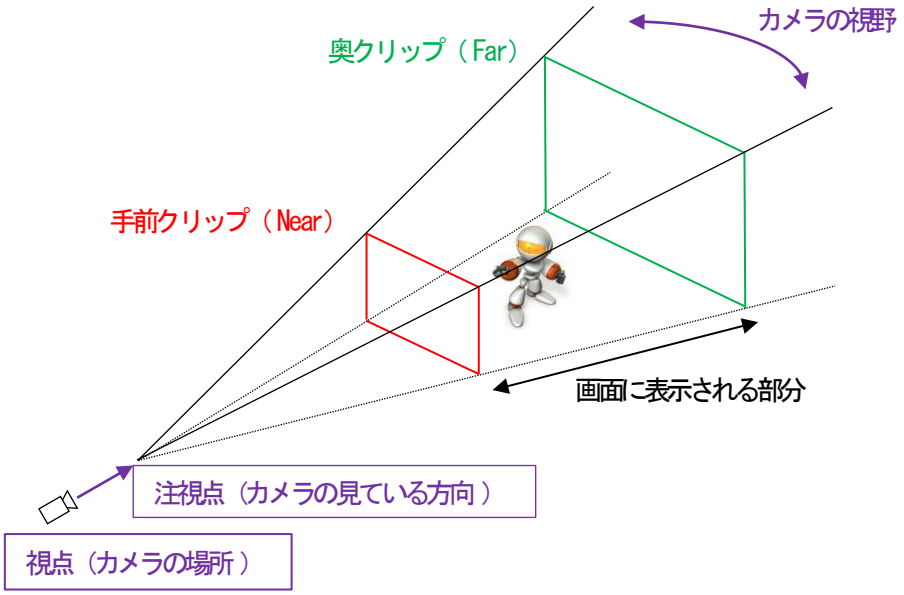
後期課題2

5日で理解する3Dプログラミング 「カメラ編」

■カメラの設定

3DCGの画面は「カメラ」そのものと言えます。カメラの見ている部分が実際の画面に反映される様になっています。



カメラ設定の基本形

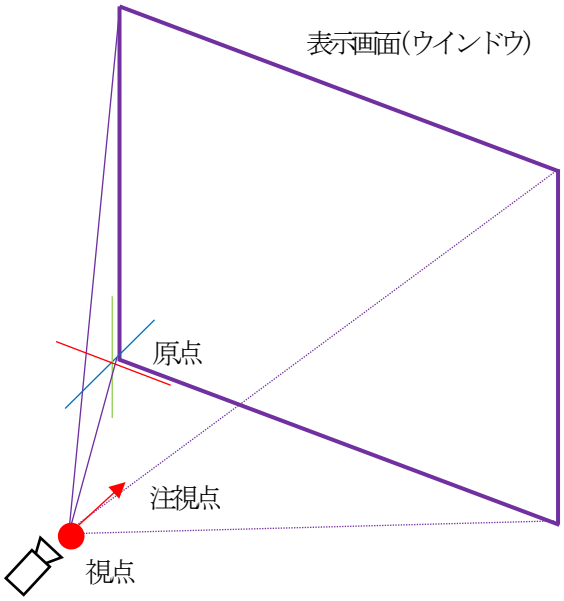
- 「カメラの座標」・・・カメラの設置位置を(x, y, z)の三次元空間の座標で設定する。
- 「カメラ上方向」・・・カメラの上方向を決定する。同じ場所でも傾けて撮影すると違いが出るイメージ。
- 「見ている方向」・・・カメラが見る方向を、注視点として(x, y, z)の座標で設定する。
- 「カメラの視野角」・・遠近法(遠いものほど小さく見える)を設定し、視野を広げたり狭くしたりする。
- 「手前クリップ」・・撮影するエリアの「一番手前」を距離(座標)で設定する。
- 「奥クリップ」・・・撮影するエリアの「一番奥」を距離(座標)で設定する。

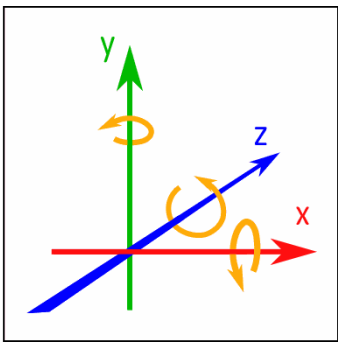
□カメラを「視点(カメラの座標)」「注視点」のみで設定する場合。

カメラの上方向がY軸で、カメラの視点、注視点を設定する
int SetCameraPositionAndTarget_UpVecY(カメラ位置, 注視点);

カメラ位置・・・・(x, y, z); // ※初期値(画面ｻｲｽﾞ X/2, 画面ｻｲｽﾞ Y/2, Y軸)
カメラの注視点・・(x, y, z); // ※初期値(画面ｻｲｽﾞ X/2, 画面ｻｲｽﾞ Y/2, 1.0f)

※デフォルト状態の表示エリアと座標系です。Z 軸の
　座標系の原点が表示画面の左下になる様に、Z 軸が画面サイズに合わせて自動計算されています。



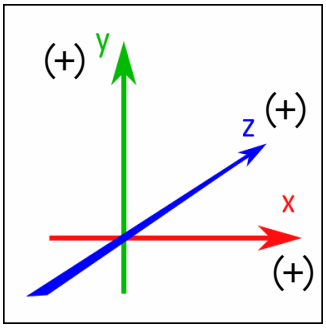


1）演習①原点の十字マークが画面の中央になるように設定しましょう。

```
カメラ視点の変数 ・・・・ VECTOR camPos;
カメラ注視点の変数 ・・・ VECTOR target;

カメラセット関数 ・・・・ SetCameraPositionAndTarget_UpVecY(視点、注視点);
```

2）演習②カーソルキーでカメラを画面の上下左右奥手前に移動できるように設定しましょう。

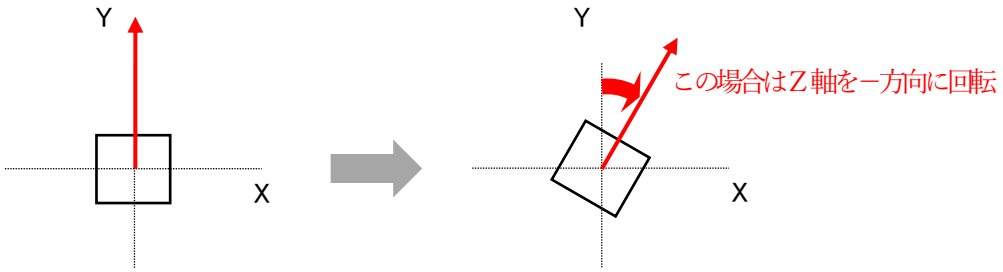
```
void Camera::Update()
{
    // ①キー入力により座標を更新する
    if(CheckHitKey(KEY_INPUT_UP)) とか
    if((KeyMng::GetInstance().newKey[P1_UP]) とか

    // ②カメラの座標と視点を常に更新する
    SetCameraPositionAndTarget_UpVecY(視点、注視点);
}
```



□カメラの上方向の設定をする場合

カメラの上方向を含めて、カメラの視点、注視点を設定する
int SetCameraPositionAndTargetAndUpVec(カメラ位置, 注視点, 上方向);

カメラ位置・・・・・ (x, y, z)
カメラの注視点・・・ (x, y, z)
カメラの上方向・・・ 上方向のベクトル (x, y, z)
※初期状態は上なので(0.0f, 1.0f, 0.0f)



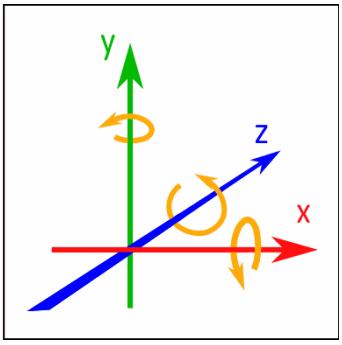
1)演習①キー入力によりカメラをZ軸回転させる。

カメラの軸回転量(ラジアン値)・・・・(x, y, z)
※それぞれの軸で回転するラジアン値を管理します。

VECTOR型にx,y,zの値を代入する場合はVGet()関数を使う

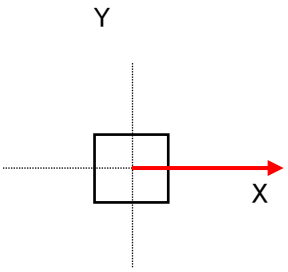
VGet(x, y, z);

例) SetCameraPositionAndTargetAndUpVec(pos,
VGet(0.0f, 0.0f, 0.0f));

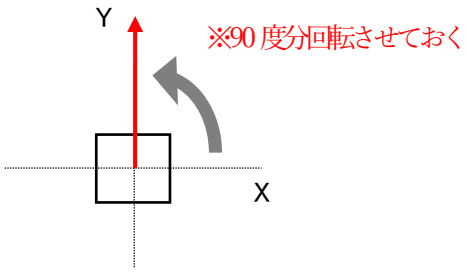


ヒント)Z軸を真っすぐ見た場合、デフォルトの角度0は時計でいう3時の方向になっているので、0の時に真上を向くように補正が必要です。

■ 初期値(0度)のZ軸の向き

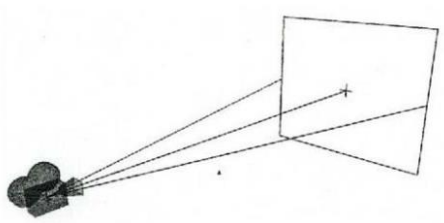


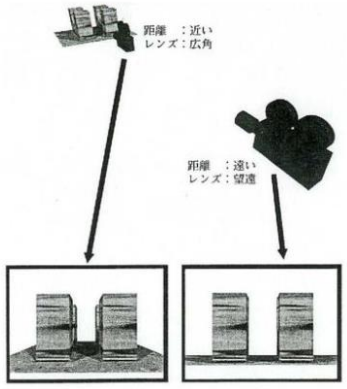
■ 0度の時に上に向く様に補正



■視野角（遠近法） Field of vision = Fov

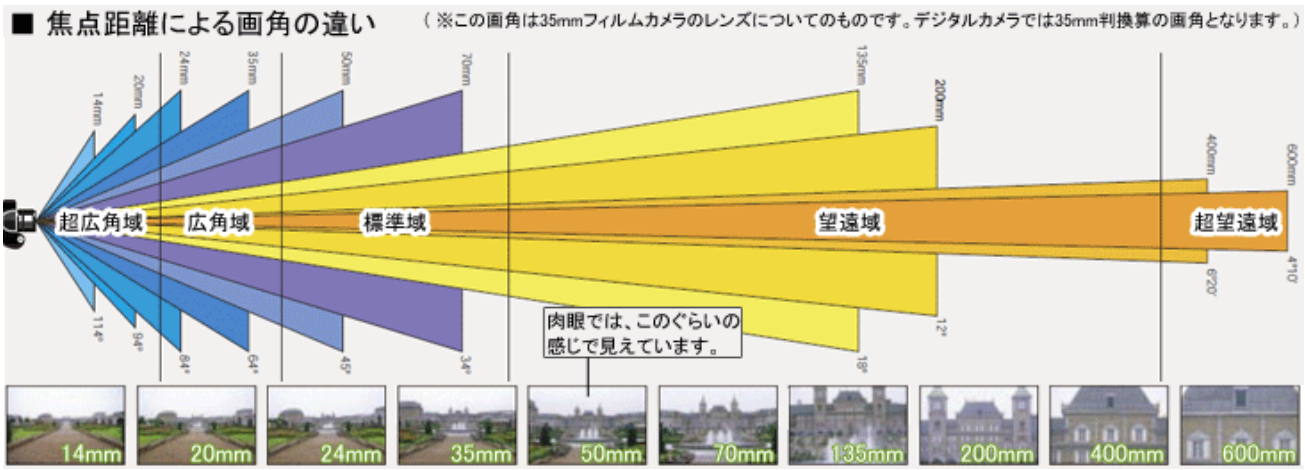
- カメラの視野（視野角）の設定
  - 対象物との距離や視野角の設定によって見え方が変わる



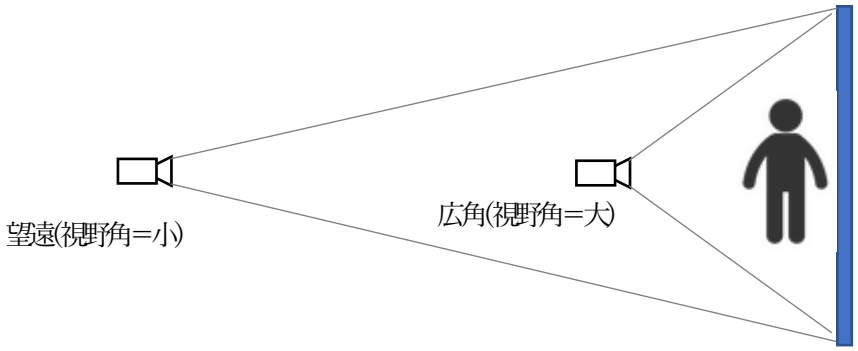


・広角 ・・・・ 実際のカメラでは焦点距離が短いとも言いますが、視野が広く多くの情報を収める事ができます。
立体感（パース）が強くかかる感じになります。

・望遠 ・・・・ 望遠と聞くと「拡大する」というイメージがありますが、視野が狭い為結果的に対象物が大きく見えます。
立体感（パース）が失われ平坦な画像になります。



画角の違うカメラで対象物を同じサイズ感で映す為には、対象物への距離が大事



```
遠近法カメラを設定する
int SetupCamera_Perspective(視野角);

視野角・・・・ float camFov; ※初期値(60 度)
                広角(最小 8 度) ＜ 60 度 ＜ 望遠(最大 160 度)
```

1)演習① キー入力により視野角の変更を行い、見え方の違いを確認する。

Camera.cpp

```
void Camera::Init()
{
   fov = (PI / 180) * 60;   // 視野(60 度)
   SetupCamera_Perspective(fov);
}
```

Camera.cpp

```
void Camera::Update()
{
  // ｷｰ入力により fov の値を変更する。
  // ※最小値(8 度ぐらい)のﾁｪｯｸを行う。
  // ※最大値(160 度ぐらい)ﾁｪｯｸを行う。

   SetupCamera_Perspective(fov);  // ｶﾒﾗｾｯﾄｱｯﾌﾟ
}
```

□視野角の無いカメラ(正射影)の設定　Orthogonal Projection

正射影は、オブジェクトを平面的に見るもので、遠近感がないので遠くのものでも近くのもので同じ大きさに見えます。
CAD などの設計において、図形を正確に表示する必要がある場合などに使用されます。

```
正射影カメラを設定する（遠いものでも近いものでも同じ大きさに見える）
int SetupCamera_Ortho(縦サイズ);

縦サイズ・・・ float Size;  ※画面垂直方向の表示範囲
```

```
void Camera::Init()
{
   float ySize = 600;
   SetupCamera_Ortho(ySize);  // 縦ｻｲｽﾞ 600 で正投影に描画する。
}
```

□カメラの「X、Y、X軸での回転」の設定

カメラの視点、垂直回転角度、水平回転角度、ひねり回転角度を設定する
int SetCameraPositionAndAngle(カメラ位置、垂直回転、水平回転、ひねり回転);

カメラ位置・・ VECTOR camPos; ※ カメラの位置
カメラ回転・・ VECTOR camRol; ※ カメラの回転
camRol.x; ※ 垂直回転(X 軸回転)のラジアン値
camRol.y; ※ 水平回転(Y 軸回転)のラジアン値
camRol.z; ※ ひねり回転(Z 軸回転)のラジアン値

Camera.cpp

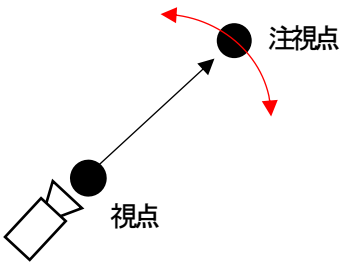
```
void Camera::Update()
{
  // 上下ｷｰ入力により camRol.x の値を変更する。X 軸回転
  // 左右ｷｰ入力により camRol.y の値を変更する。Y 軸回転
  // ctrl ｷｰ入力により camRol.z の値を変更する。Z 軸回転

   SetupCameraPositionAndAngle(camPos,
                camRol.x, camRol.y, camRol.z);  // ｶﾒﾗｾｯﾄｱｯﾌﾟ
}
```

■FPS っぽいカメラの再現

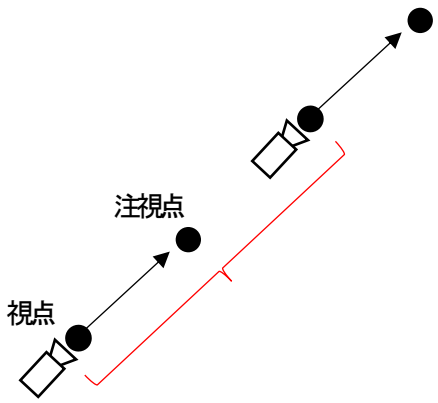
カメラ移動の集大成として FPS の自分視点カメラを作成します。
「A」「D」キーで左右に回転し「W」「S」キーで向いている方向に前後移動する様なカメラを設定しましょう。

1)カメラの仕様の確認

①視点の回転に合わせて注視点を回転させる。



②向いている方向に移動する。

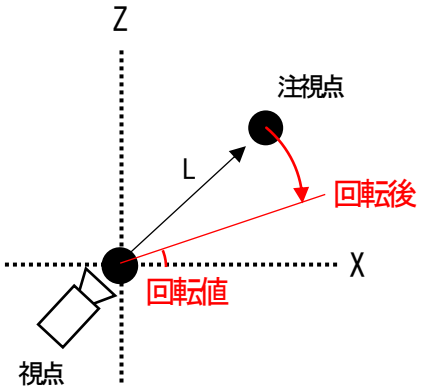


2)計算方法

必要な変数

```
VECTOR camPos;  // 視点 (camPos.x, camPos.z)
VECTOR target;  // 注視点(target.x, target.z)
VECTOR camRol;  // Y軸の回転角 = camRol.y(ラジアン)
float L;        // 注視点までの距離(単位ベクトル可)
```

①視点の回転に合わせて注視点を回転させる。(Lの距離分を加味した回転後の座標)

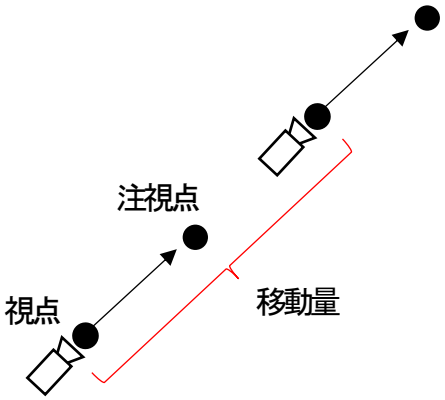


回転後の角度 = 現段階での回転値
+ 回転角度(＋値or－値)

回転後X = camPos.x + cos(回転後の角度)＊L
回転後Z = camPos.z + sin(回転後の角度)＊L

※回転後の座標がtarget(注視点)となる事で、カメラが
見ている方向を指す事になります。

②向いている方法に移動させる。



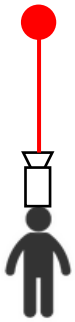
回転値 = 回転後の角度

移動後X = camPos.x + cos(回転値)＊移動量
移動後Z = camPos.z + sin(回転値)＊移動量

※回転後の角度の方向にX、Z軸での増減分で移動する。
2Dゲームの斜め移動と原理は同じです。

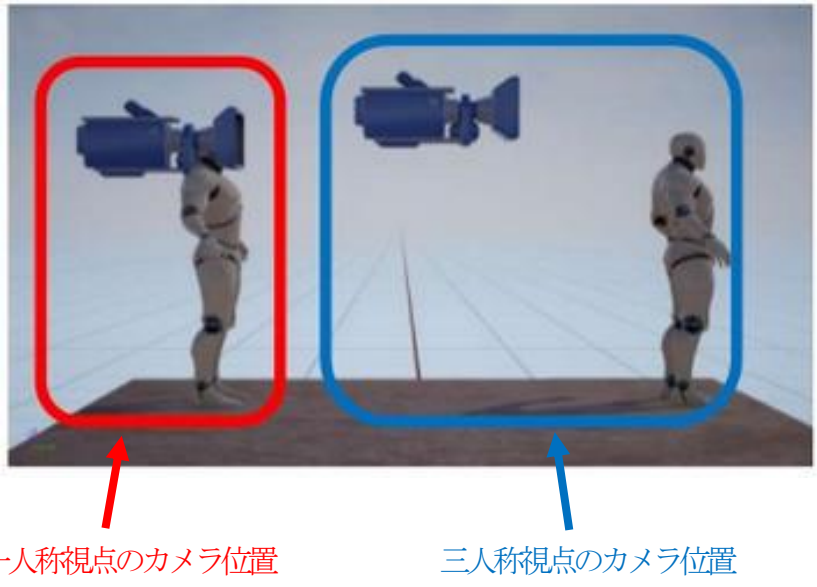
③一人称と三人称視点の考え方。

1）一人称視点の場合



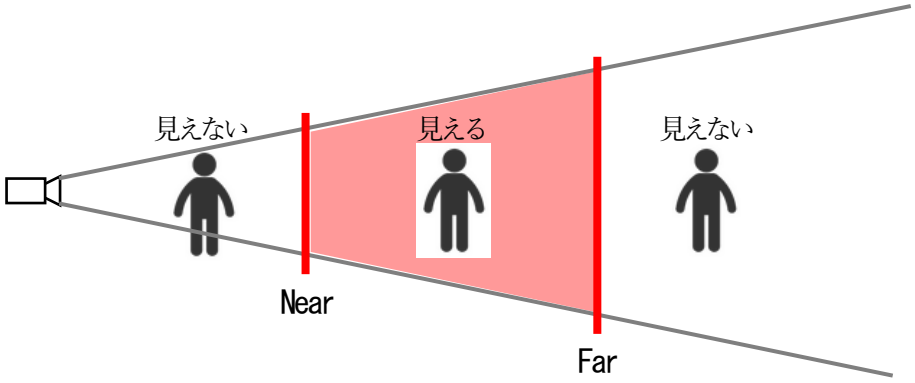
2）三人称視点の場合

3）イメージ図



□カメラの「手前」「奥」の限界点の設定

カメラに映る範囲(奥行)は、Near(ニア)とFar(ファー)の間のエリアとなります。
このエリアの事を「視錐台」と言い、視錐台の外のものは描画されません。
NearとFarで手前と奥の限界点を設定でき、エリアを広げると広大な世界を構築できますが、
その分、処理が増加して描画などの負荷がかかりますので、程よい範囲設定が必要となります。
尚、NearとFarで切り取られたエリアを「クリップ」「クリッピング」と言います。



カメラの 手前クリップ距離と 奥クリップ距離を設定する
int SetCameraNearFar(手前クリップ距離, 奥クリップ距離);

手前クリップ距離・・float camNear; ※0.0f より大きく Far より小さな値
奥クリップ距離・・・float camFar; ※Near より大きな値

camNear の最小値は 1.0f ぐらいが妥当

参考例

```
// ---------- カメラ1
//奥行10.0～1000までをカメラの描画範囲とする
camNear = 10.0f;    // 手前
camFar = 2000.0f;   // 奥
SetCameraNearFar(camNear, camFar);  // 初期セット
```

■まとめ「カメラの基本知識」

・「カメラの座標」・・・カメラの設置位置を（x, y, z）の三次元空間の座標で設定する。
・「カメラ上方向」・・・カメラの上方向を決定する。同じ場所でも傾けて撮影すると違いが出るイメージ。
・「見ている方向」・・・カメラが見る方向を、注視点として（x, y, z）の座標で設定する。
・「カメラの視野角」・・遠近法（遠いものほど小さく見える）を設定し、視野を広げたり狭くしたりする。
・「手前クリップ」・・撮影するエリアの「一番手前」を距離（座標）で設定する。
・「奥クリップ」・・・撮影するエリアの「一番奥」を距離（座標）で設定する。