「　座標　」

東京情報クリエイター工学院

座標

◆座標

１．２Ｄの世界（デカルト数学）

おそらく多くの方が、デカルトという単語を今まで聞いたことがないとしても、必ず 2D デカルト座標系は使ったことがあるでしょう。地図、黒板、オセロ、将棋を知っていれば、2D デカルト座標系に触れていることになります。

すべての 2D デカルト座標空間は以下のものを持ちます。

・原点と呼ばれる座標系の中心を表す特別な場所を持つ。
・原点を通る軸と呼ばれる 2 つの直線を持つ

２．座標

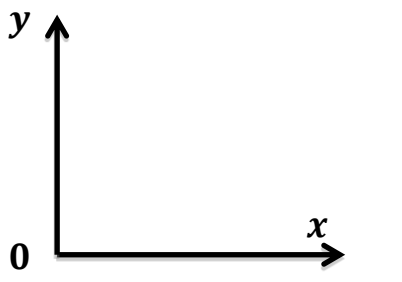
2D デカルト座標空間では、「平面に直交する 2 つの直線(座標軸)と x 軸と y 軸を定め、平面上の点に対してそれぞれの座標 と x,y を組にして表したものをその点の座標」といいます。

1 組の座標を書くときに使われる標準的な書き方は，数字と「 （ ） 丸括弧」を使って書きます。

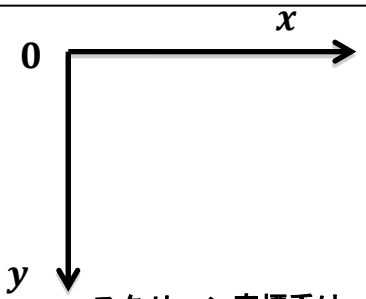
2D 座標空間・・・2 数の組
3D 座標空間・・・3 数の組

今まで学習してきた座標の表現方法はデカルト座標系です。コンピュータで使う座標の表現法は違う場合があります。

例えば、コンピュータのスクリーン上の画像を扱うときは慣習的に次のような座標系（**スクリーン座標系**）を使います。



通常の座標系は[　　]が原点となる



スクリーン座標系は[　　]が原点となる

３．２Ｄから３Ｄへ

2D デカルト空間がどのように機能するかは中高の授業で習ってきましたね。平らな 2D の世界を離れて、3D 空間について考えると 3D 空間は 2D 空間よりちょっとだけ複雑に感じるでしょう。3D はたった 1 つの次元が多いだけであり、今まで 2 つの次元は学習し知識を持っていますからね。

しかし残念なことに、そうではないのです。さまざまな理由から、3D 空間は 2D 空間よりも可視化したり記述したりすることがずっと難しいのです。この難しさの原因の 1 つが、本やコンピュータのスクリーン上の図は 2D であるのに対して、我々の物理的な世界は 3D です。2D で解くのが容易な問題が、3D では格段に難しくなることや、また「未定義」であることがよくあります。それでも「**2D の多**

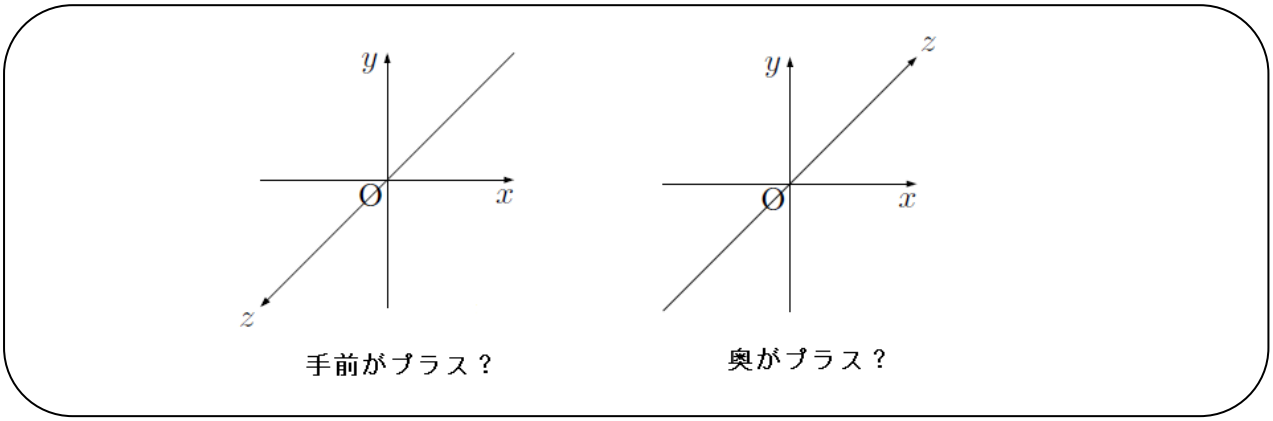
「　座標　」

東京情報クリエイター工学院

くの概念は 3D に直接拡張でき」多くの場合、2D を用いて問題を理解し、解法を作り出し、その解法を 3D に拡張するのです。

3D では、座標系を作成するために 3 つの軸を必要とします。はじめの 2 つの軸は 2D と同様に、x 軸と y 軸で、3D 空間に拡張するのに 3 つ目の軸 z 軸を追加します。2D では、x 軸と y 軸は互いに直交していました。3D でも同様に、それぞれの軸は互いに直交します。

では、 z 軸を追加したときに z 軸のプラス方向はどちらに向くのでしょうか？



４．右手座標系と左手座標系

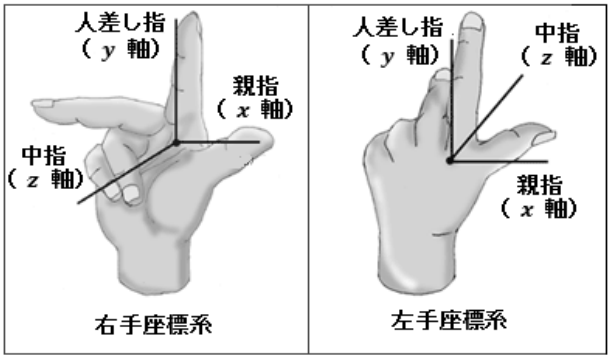
3D 座標空間に z 軸の方向は値を求める際にとても重要です。

(1)右手座標系

右手の親指を右（x 軸の正の向き），人差し指を上に（y 軸の正の向き）に向けると、中指は前方（z 軸の正の向き）を向きます。

(2)左手座標系

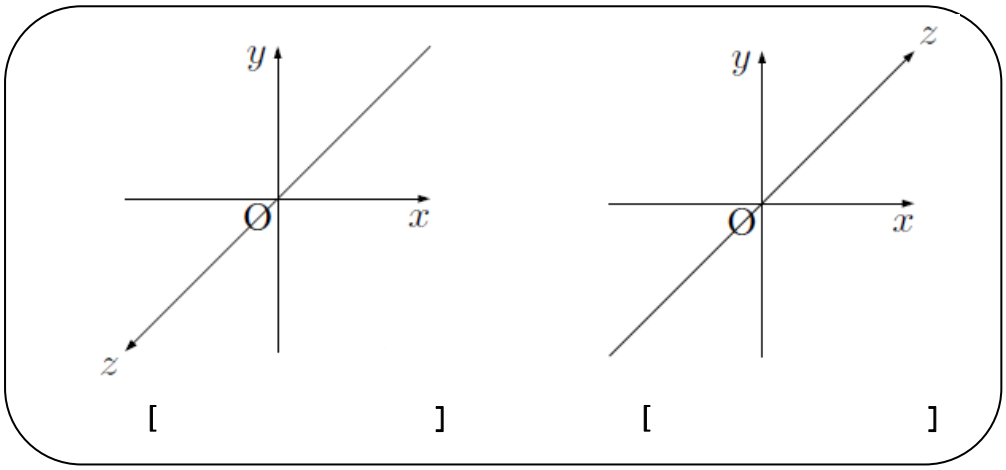
左手で同じことをやると，z 軸が後方を向きます。



※3D に関する教科書・参考書を読む場合、どちらの座標系でかかれているかしることが大切！

メジャーなグラフィックライブラリとして、DirectX と OpenGL の 2 種類があります。このグラフィックライブラリは座標系が異なります。

さて、右手系・左手系どっちがどっち？



従って、グラフィックライブラリを利用してプログラムする場合、これら座標系を考慮する必要があります。

５．複数の座標系

ゲームプログラミングにおいては、管理のしやすさや目的に応じて、次の座標系を使います。

(1) ローカル(モデル)座標系
↓　ワールド座標変換
(2) ワールド座標系
↓　ビュー座標変換
(3) ビュー(カメラ)座標系
↓　射影座標変換
(4) 射影(プロジェクション)座標系
↓　スクリーン座標変換
(5) スクリーン座標系

これらの座標系を使い分けながらプログラムを組んでいきますが、最終的には射影座標系に変換してディスプレイに出力します。座標の変換には、マトリクス(行列)が利用されます。

(1) ローカル(モデル)座標系

ゲームでは、必要に応じてプレイヤーや敵、背景のオブジェクトをゲームの世界に配置します。それぞれのオブジェクトには、形状がありますが、**ローカル座標系**は、そのオブジェクト単体の座標を管理するためのものです。オブジェクトごとにローカルな座標系を持たせることで、ゲームの世界に配置しやすくなります。

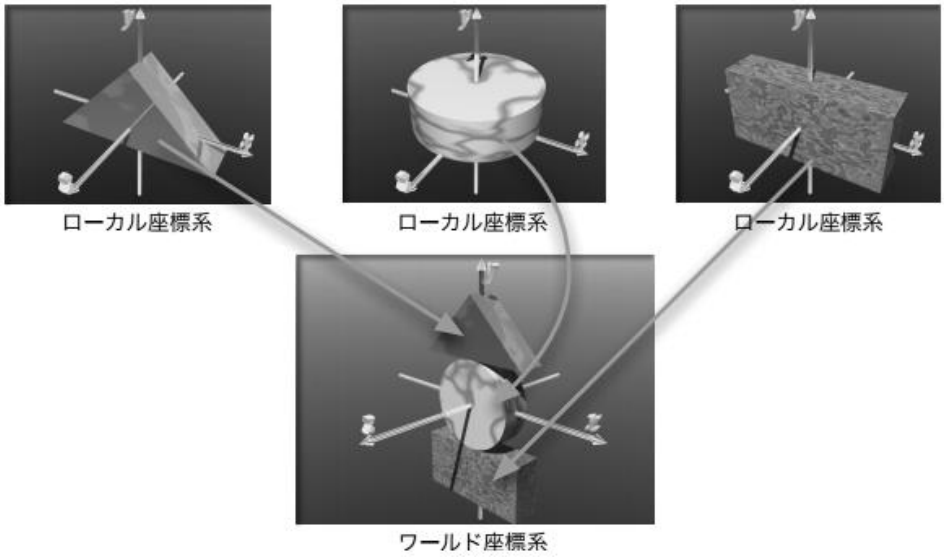
(2) ワールド座標系

各オブジェクトは、実際のゲームの世界に配置されなければなりません。この「実際のゲームの世界」の座標系のことをワールド座標系といいます。つまり、**オブジェクトを任意の場所に配置**したりするのが**ワールド座標系**です。

各オブジェクトをローカル座標で管理したのは、それぞれを配置しやすくするためです。というのは、マトリクス１つあれば、それぞれのオブジェクトをワールド座標系に簡単に配置できるからです。

つまり、まず、オブジェクトをローカル座標で組み立てます。 それをワールド変換行列でローカル座標からワールド座標に変換します。このときワールド変換行列は平行移動行列、回転行列、スケーリ

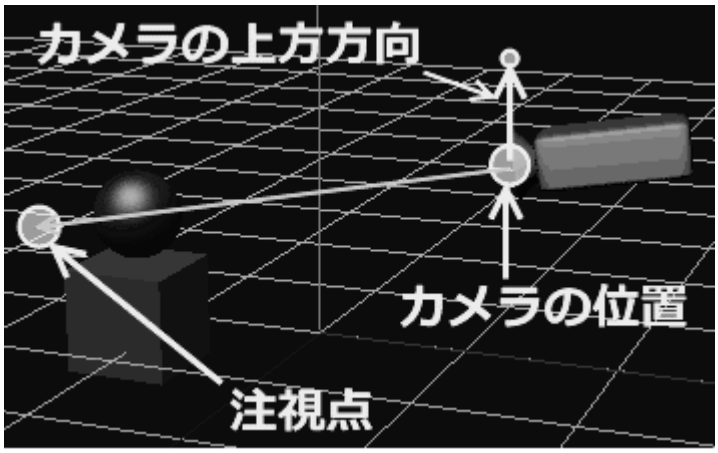
ング行列を使っていきます。



(3)ビュー(カメラ)座標系

(2)で各オブジェクトをワールド座標系に配置しました。ですが、このままだとどこから物体を見ているのかがわかりません。私たちはGPSのようなワールド座標系で物事を見ているのではなく、各人の視点から見てますよね。従って、今度は各オブジェクトを「どこから見て」「どこを見ている」のかという情報が必要になります。そこで、ワールド座標系にカメラを配置します。
このカメラは、現実世界のカメラと同じで、風景や動いているものを撮影します。そして、その像を画面に投射することになります。

ビュー座標ではカメラの位置を定義することによって物体をビュー座標にすることができます。カメラの位置の定義としては**カメラの位置、カメラの注視点、そしてカメラの上方方向**があります。なぜカメラの上を定義するのかと言いますと、テレビを寝転んでみるのと正座してみるのとでは違いますよね。 カメラも同様で上がはっきりしないと全く違った出力になります。

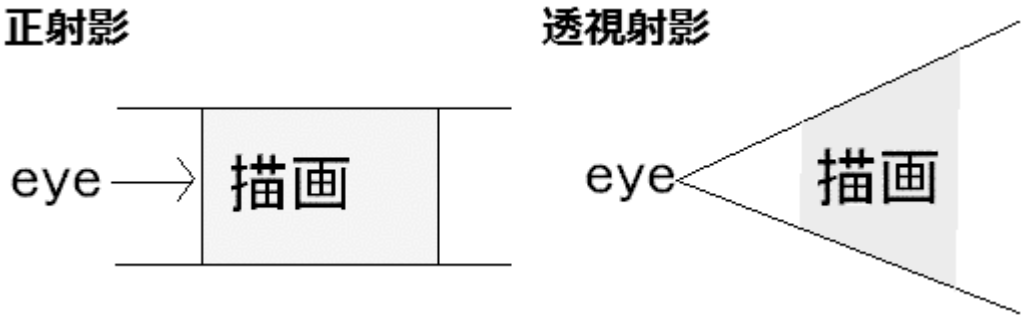


ビュー座標系への変換は、ワールド座標変換と同じことをします。 ワールド座標変換は定義された位置まで平行移動しました。ビュー座標変換は逆に、カメラの座標を定義された位置から原点まで移動します。 そしてその負の移動座標(逆マトリクス)を全てのオブジェクトに適用します。 そうすることでカメラを中心とした座標ができます。

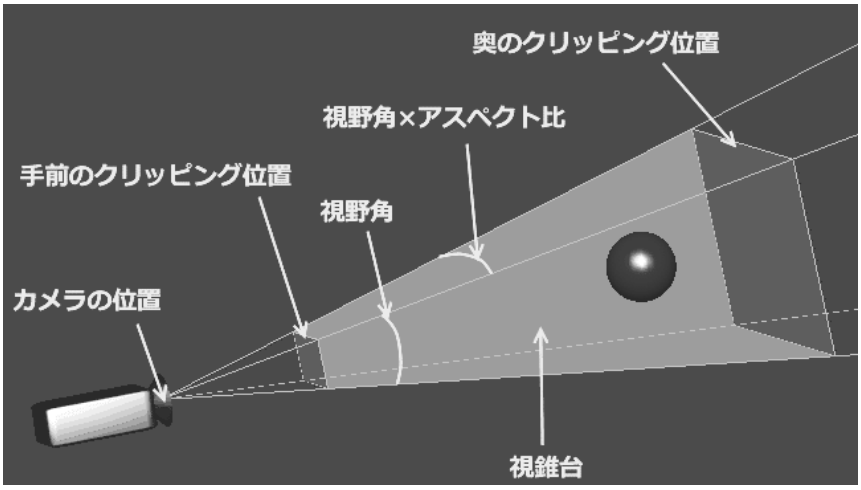
(4) 射影(プロジェクション)座標系

先ほどはカメラの位置などを求めました。 ですが、実際の世界はズームなどの機能もありますね。同様にプログラムの世界もカメラの設定をしなくてはなりません。 このカメラの設定をするのがこのプロジェクション座標変換行列です。プロジェクション座標変換行列には二通りあります。 一つは「**正射影**」、もう一つは「**透視射影**」です。

正射影は視点の手前側にあるのと視点の奥行きにあるものの大きさは同じ大きさになります。左の図のような視点となっているためです。それに対して透視射影は、実際にカメラを覗いてるように物体が視点に近いと大きくなり、遠いと小さくなります。

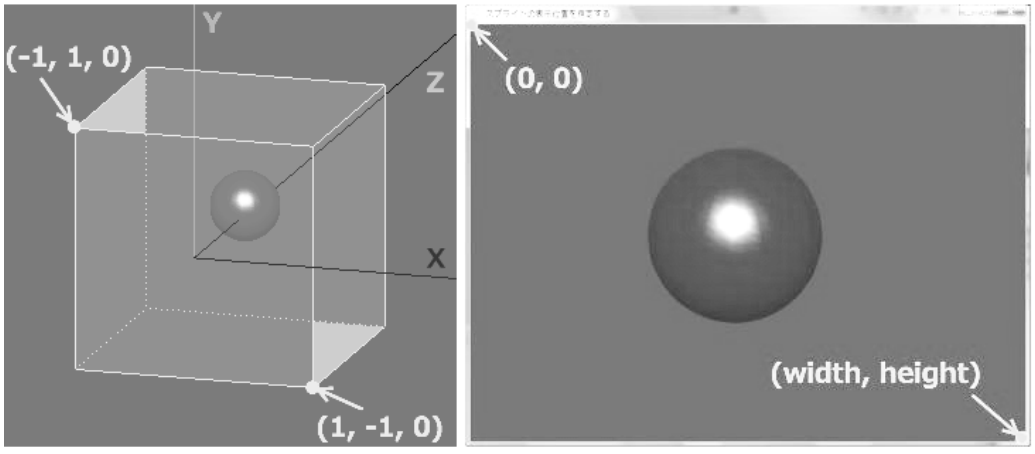


一般的に使われる「透視射影」のイメージは下のようになります。



(5) スクリーン座標系

射影変換を行った後、実際のスクリーンの座標に変換します。



「　座標　」

東京情報クリエイター工学院