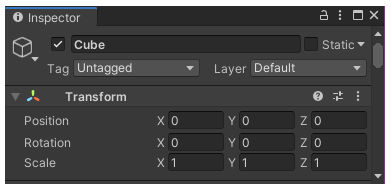
ベクトル・行列応用②　～平行移動～

**■回転操作以外の変換行列（Transformation Matrix）**

「ベクトル・行列応用①」ではベクトルの回転（座標系の回転）について調べ、正規直交基底ベクトルの組合せが座標系になっており、さらに回転行列にもなっていることがわかりました。　ここでは、あるベクトルを別のベクトルに変換する行列について、回転操作以外の変換行列を紹介します。

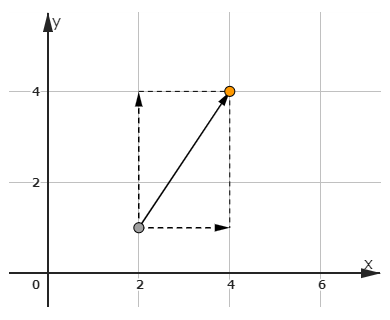
**■平行移動（トランスレーション）行列**

Unityでゲームオブジェクトのインスペクターを見てみると、右図のように【Transform】というコンポーネントがあります。 じつは内部では行列のデータになっています。 このセクションでは平行移動（右図の【Position】）の部分を見ていきたいと思います。 まずはUnity上で動きと行列データの確認をしてみましょう。 平行移動は親オブジェクトの座標系上で移動していることがわかります。



※Positionは「位置」という意味ですが、

座標系の「原点から平行移動した場所」と同義です



さて、我々は座標系を行列で表現できること、任意のベクトルの回転操作は行列との積で可能であることを知っています。 そこで、回転操作以外の座標変換も行列との積で可能にしたいと考えます。 その理由としては、

のように、複数の座標変換を事前に行列の積で求めておけば、大量のポリゴン頂点の座標変換コストを抑えることが可能にできるからです。 また、行列との演算で統一することで、処理の流れをシンプルにすることができるからです。

よりも

それでは、すべての平行移動の動きをベクトルと行列との積で表現してみましょう。

よって

をどうやら決められそうにありません。

しかしながら、いろいろな座標変換を行列とベクトルとの積で計算させることが目的なので、どうにかしたいところです。

**★同次座標系（Homogeneous Coordinates）に拡張**

さきほどの計算式をもう一度見てみましょう。

ここで、

となり、 式のそれぞれの両辺に を足せば、 右辺が の形になります。

何か見えてきませんか？　左辺だけに着目してみましょう。

なんと！ 式は

とおいたときの　の 成分、 成分になっていることがわかります。

どうやら、次元数を1つ上げてあげれば計算可能なようです。　その次元数を上げた空間を

【同次座標系（Homogeneous Coordinates）】とよび、

ゲームエンジンやグラフィクスライブラリの中身では、この同次座標系で処理がおこなわれています。

操作したい位置ベクトルや方向ベクトルの次元数を上げた成分は、基本的に1にします。

ほかの値にすることもできますが、ここでは取り上げません。　興味がある方は下記リンク先を参考にしてください。

* <https://xr-hub.com/archives/12124>
* <http://marupeke296.com/DXG_No55_WhatIsW.html>

ところで、 式に着目してみると、次元数が1落ちてしまっていることに気づきます。　これでは問題があります。 どのような問題があるのでしょうか。入力は3次元ベクトルなのですが、出力が2次元ベクトルになっていますので、つぎの座標変換の計算ができません。

（例）異なる平行移動を2回おこなう場合

計算可能にするにはどうしたらいいでしょうか。

じつは、平行移動行列 を3次の正方行列に拡張すれば問題なく計算できるようになります。

あらためて , とおいて、　となる 　を見つけます。

を常に満たすには、 であればよいので

となりました。

※今後は同次座標系で考えるので、基本的に 記号は省略します

**■回転行列も同次座標系へ拡張**

平行移動の操作は、対象のベクトルと変換行列を同次座標系に拡張すればよいことがわかりました。　そして、回転行列についても同次座標系に拡張する必要があります。

（例）これまでの回転行列では計算できない

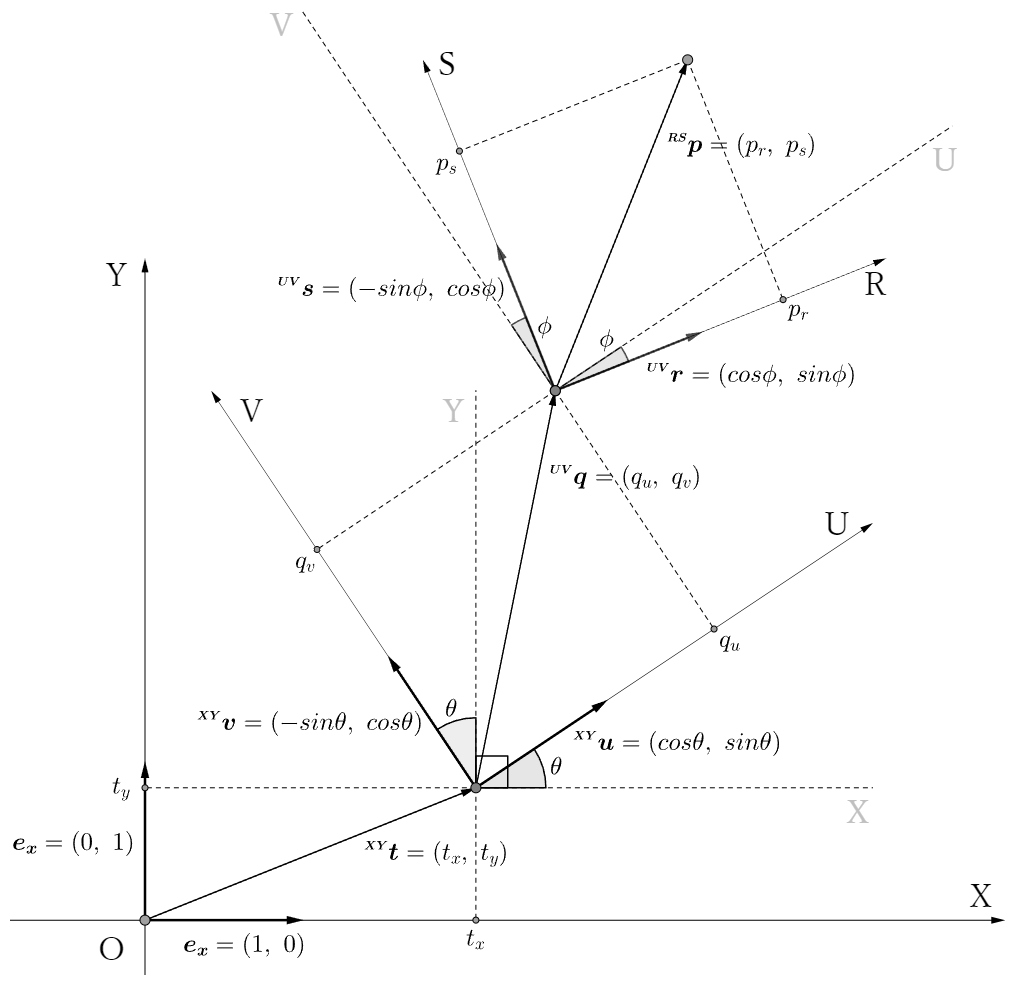
結果的に以下のような回転行列になります。

**■ローカル座標系に位置情報を追加しよう**

最初にUnity上で動きと行列データの確認をしました。　ゲームオブジェクトのローカル座標系は、そのオブジェクトの

* 親座標系上における姿勢（基底ベクトルの向き）
* 親座標系上における位置

を決定します。　下図のような状態です。



ところで、ゲームエンジンやグラフィクスライブラリでは、それぞれのローカル座標系におけるベクトルを最終的にはワールド座標系に変換する必要があります。　なぜなら、画面に表示するために以下の流れで座標変換をおこなうからです。

**ワールド座標変換**

**ビュー座標変換**

※ビュー座標系とは、

カメラ基準の座標系

**プロジェクション座標変換**

**スクリーン座標変換**

各ローカル座標の座標

（同次座標系のベクトル）

ワールド座標系の座標

ビュー座標系の座標

プロジェクション座標系の座標

スクリーンのピクセル座標

ここで を例にして、 に変換することを考えます。

まず

1. の原点と一致している場合（その座標系を座標系とする）を考え、

、は次式で求められます。

ただし、

1. ①で求めた　を

平行移動行列を

ただし、

①、 ② より

したがって、

と表現できることがわかりました。　じつはこの **、** Unity　におけるゲームオブジェクトのTransform内部の4 x 4行列データとなっているのです（今回は2次元で考えているので3 x 3行列）。　一般に【モデル行列】とよばれています。

ここで　を計算してひとつの行列にまとめます。

他によくみる表現としてはがあります。

ただし

まとめ