1 Szenario

Ziel dieses Tutorials ist es,den Umgang mit Fundamentalrotationen zu erläutern und schließlich den Zusammenhang zur Yaw-Pitch-Roll Matrix herzustellen.
Eine Fundamentalrotation ist eine Rotationsmatrix R, die eine Rotation um eine der 3 Koordinatenachsen ausübt.Warum diese Fundamentalrotationen so wichtig sind zeigt das Theorem von Euler:

> *Zwei voneinander unabhängige, orthonormale Koordinatensysteme können durch eine Folge von nicht mehr als 3 Fundamentalrotationen, wobei keine zwei aufeinanderfolgenden Rotationen um die selbe Achse erfolgen, ineinander überführt werden.*

Die Winkel, um die dabei rotiert wird, werden Eulerwinkel genannt. Die Fundamentalrotationen sind wie folgt Charackterisiert:

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{pmatrix}$$

$$R_y(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}$$

$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In der Bibliothek lassen sich die Matrizen wie folgt erstellen:

```
Matrix3D Id,R_x,R_y,R_z;
double alpha,beta,gamma;

R_x = Id.rotateX(gamma);
R_y = Id.rotateY(beta);
R_z = Id.rotateZ(alpha);
```

Wer direkt seinen Vektor um eine der Koordinatenachsen rotieren möchte, für den gibt es noch folgende Möglichkeit dies zu tun:

```
Vector3D s(1,2,3);
Vector3D rotated;
double alpha,beta,gamma;

rotated = rotateX(s,gamma); // fuer Rotation um x
rotated = rotateY(s,beta); // fuer Rotation um y
rotated = rotateZ(s,alpha); // fuer Rotation um z
```

2 Yaw-Pitch-Roll Matrix

Nachdem jetzt bekannt ist, dass 2 Koordinatensysteme durch maximal 3 Rotationen ineinander überführt werden können, definieren man sich:

- Yaw γ : Rotation gegen den Uhrzeigersinn um die z-Achse
- Pitch β : Rotation gegen den Uhrzeigersinn um die y-Achse
- Roll α : Rotation gegen den Uhrzeigersinn um die x-Achse

Nach dem Eulertheorem stehen nun 12 Möglichkeiten zur Verfügung, die eindeutig 2 Koordinatensysteme ineinander abbilden:

- xyz yzx zxy
- xzy yxz zyx
- xyx yzy zxz
- xzx yxy zyz

Wobei xyz beispielsweise bedeutet: zuerst erfolgt eine Rotation um die x-Achse als nächstes eine Rotation um die neue y-Achse und zuletzt eine Rotation um die neue z-Achse. In der Luftfahrt hat man sich auf die ZYX-konvention geeinigt. Also zunächst erfolgt eine Rotation um die Z-Achse (Yaw), dann erfolgt eine Rotation um die neue Y-Achse(Pitch) und schließlich eine Rotation um die X-Achse(Roll).

Die Yaw-Pitch-Roll Matrix ist nun genau diejenige Matrix, die entsteht,wenn man die Fundamentalmatrizen der Reihenfolge(zyx) von rechts miteinander multipliziert, also:

$$R(YPR) = R_z(\alpha) \cdot R_y(\beta) \cdot R_x(\gamma) =$$
$$\begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}$$

Da es sehr mühselig ist, sich all dies bei jeder Verwendung vor Augen zu halten, kann dies mit der Bibliothek relativ bequem erzeugt werden

```
double yaw =M_PI/4;
double pitch =M_PI/6;
double roll =M_PI/12;

YPR y(yaw,pitch,roll)
Matrix3D YPR = y.toMatrix3D();
```