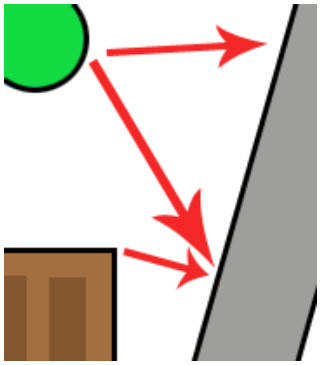
0.0.1 Komponent 3: Bestem placering af Transportbånd

For at kunne bestemme placeringen for transportbåndet skal der udføres nogle beregninger vha. 3 målinger fra brugeren til transportbåndet. Disse ses på figur 1



Figur 1: Oversigt over målinger til at beregne transportbåndets placering

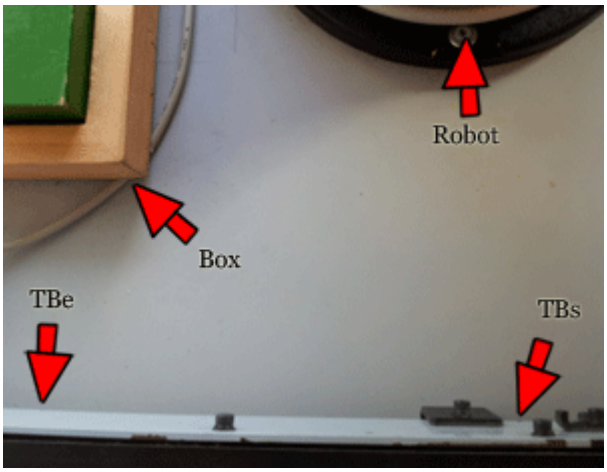
Med de tre længder kan der beregnes nogle relativt præcise koordinater for, hvor robotten skal samle klodsen op på transportbåndet, og hvor den skal sætte den på transportbåndet for at rotere den.

De 3 punkter, *BTBe*(Box-TransportBåndEnde), *RTBe* (Robot-TransportBåndEnde) og *RTBs* (Robot-TransportBåndStart), har bestemte punkter der skal måles mellem.
Eftersom robottens centrum er defineret med koordinatsættet *(0;0)* i robottens indre, er det ikke muligt, at foretage en måling dertil.
Derfor er punktet, *Robot* valgt ved robottens fod.
For *Box* er punktet ved kassens hjørne valgt.
For *TransportBåndEnde* (*TBe*) er et punkt under et hul på transportbåndet valgt.
For *TransportBåndStart* (*TBs*) er et punkt ved siden af transportbåndets fod valgt.

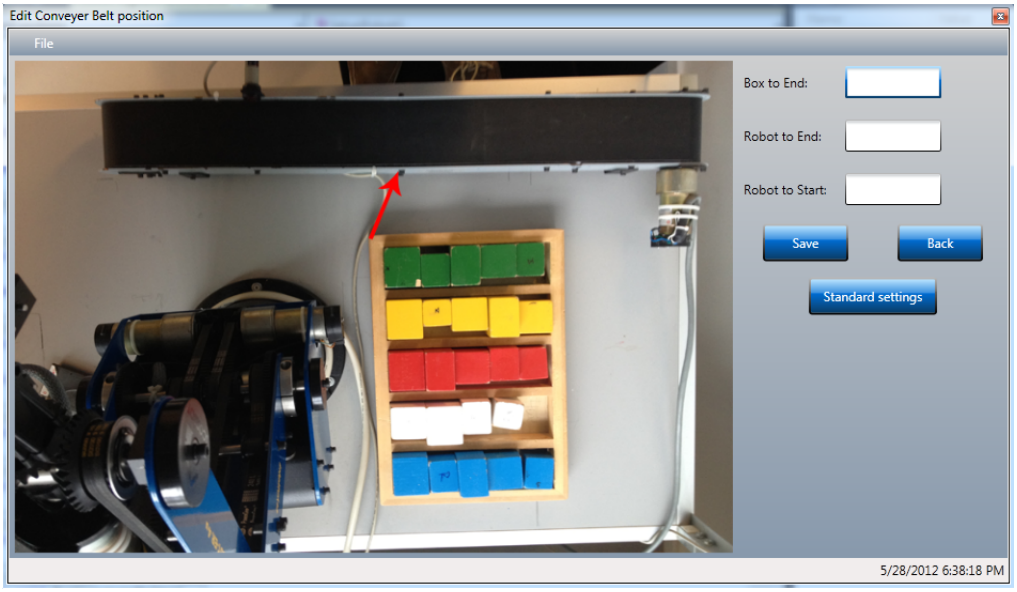
De følgende længder, $|BTBe|$, $|BTBe|$ og $|BTBe|$, skal måles og kan angives i vinduet, vist på figur 3.

For bedre overskuelighed vises målingerne i et koordinatsystem på figur 4.

De tre stiplede linjer på figur 4 er de tre sider der skal måles. De 2 hele linjer er længden mellem robotten og kassen, samt 2 punkter på transportbåndet.



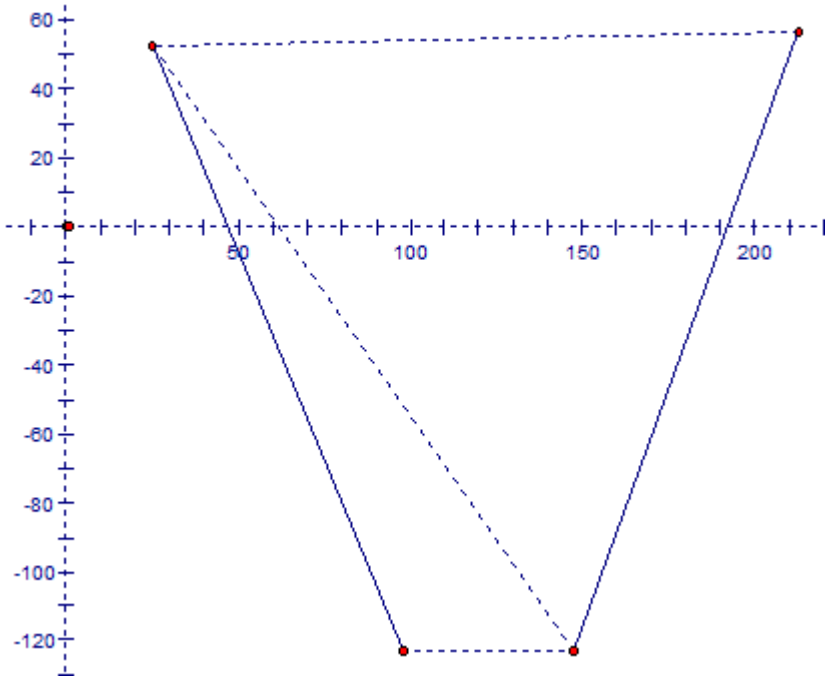
Figur 2: Punkter der måles imellem



Figur 3: Vindue måling skal tastes ind i

Eftersom transportbåndet kan placeres i en vilkårlig position, er der dog nogle begrænsninger:

1. Højden må ikke justeres på nogen måde.
2. Transportbåndet skal være placeret på den viste side, som vist på figur 2.
3. Transportbåndet må ikke roteres til siden.
   (a) Det må ikke ligge på siden.
   (b) Det må ikke tilte.



Figur 4: Oversigt over mulig opstilling for transportbåndet

4. Transportbåndet må ikke roteres rundt, således båndet kører fra robotten mod kassen.

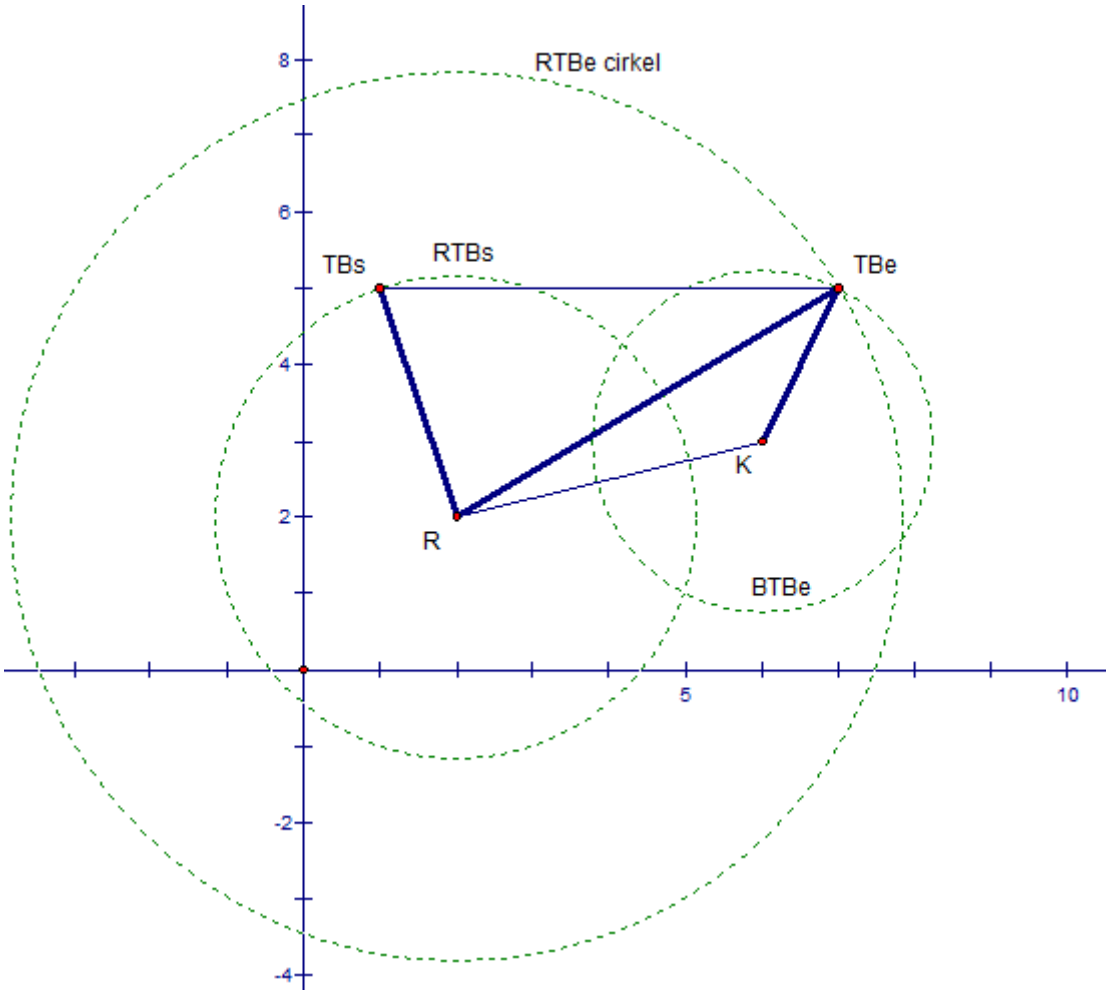
BEREGNINGER

For at beregne hvor transportbåndet er, i forhold til robotten og kassen, samt hvor klodsen skal samles op og roteres, kræver en del beregninger. Disse vises herunder:

Når de 3 målinger er indtastet, eksekveres følgende metode:

Kodeudsnit 1: Funktionen CalculateTBCoords(...)

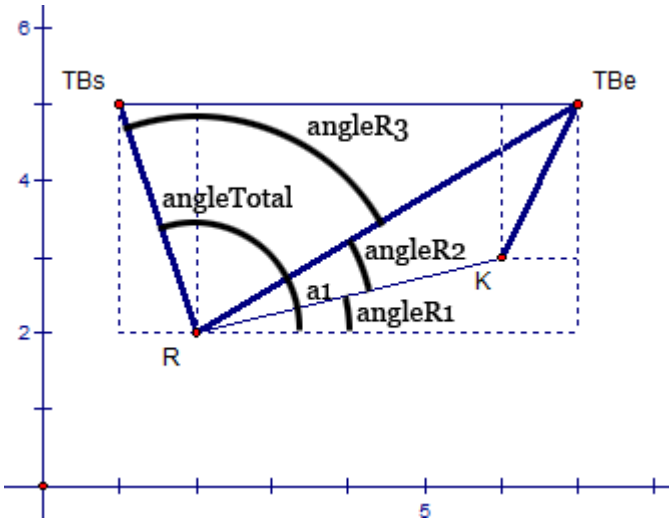
```
1 internal void CalculateTBCoords( out double xTBe, out double yTBe↩
    , out double xTBs, out double yTBs )
2 {
3   double a1;
4
5   a1 = (yKasse - yRobot) / (xKasse - xRobot);
6   if (double.IsNaN(a1))
7   {
8       a1 = 120; //Hældning på 120 er meget stejlt og vil gå an
9   }
10
```



Figur 5: Beregninger for Transportbåndet. De fede streger viser målinger og cirklerne viser hvor længden kunne være

```
11    double angleR1 = Math.Atan(a1) * deg;
12
13    double angleR2 = CosEquation(KTBe, RTBe, RK);
14    var angleTotal = angleR1 + angleR2;
15
16    yTBe = yRobot + RTBe * Math.Sin(angleTotal*rad);
17    xTBe = xRobot + RTBe * Math.Sin((180 - 90 - angleTotal)*rad);
18
19    double angleR3 = CosEquation(TB, RTBe, RTBs);
20    angleTotal += angleR3;
21    yTBs = yRobot + RTBs * Math.Sin(angleTotal*rad);
22    xTBs = xRobot + RTBs * Math.Sin((180 - 90 - angleTotal) * rad);
23 }
```

Beregningerne forklares bedst med en tegning:



Figur 6: Illustration af beregninger

Først beregnes hældningen på mellem kassen og robotten (linje 5):

$$a = \frac{y_2 - y_1}{x_2 - y_1} \tag{1}$$

$$a1 = \frac{y_{kasse} - y_{Robot}}{x_{kasse} - x_{Robot}} \tag{2}$$

Det er muligt, at ændre koordinaterne for kassen eller robotten, således de står "lige over hinanden", altså deres x-værdi bliver ens. Hvis dette er tilfældet vil der blive divideret med nul.

Dette er normalt en ulovlig operation, men da *xKasse* og *xRobot* er defineret af typen *double*, vil værdien være *NaN* - uendelig stor.

Skulle det være tilfældet, kan der blot indsættes en meget stor hældning som erstatning, da "uendelig stor"ikke er til at regne med.

Herefter beregnes hældningen i grader (linje 11):

$$\theta = tan^{-1}(a) \tag{3}$$

$$angleR1 = tan^{-1}(a1) \cdot \frac{180}{\pi} \tag{4}$$

Der ganges med $\frac{180}{\pi}$, da værdien ønskes gemt i grader og C#'s *Math*-klasse bruger radianer.

Herefter beregnes vinklen $\angle BoxRobotTBe$ med cosinus-relationen (linje 13):

$$\angle A = cos^{-1}\left(\frac{b^2 + c^2 - a^2}{b \cdot c \cdot 2}\right) \tag{5}$$

$$angleR2 = cos^{-1}\left(\frac{|RTBe|^2 + |RK|^2 - |KTBe|^2}{|RTBe| \cdot |RK| \cdot 2}\right) \tag{6}$$

De to vinkler ligges herefter sammen, således den fulde vinkel til TBe findes (linje 14):

$$angleTotal = angleR1 + angleR2 \tag{7}$$

Nu kan y-koordinaten til TBe findes vha. sinus-relationen, samt $Robot$'s x- og y-koordinat (linje 16-17):

$$\frac{a}{sin(A)} = \frac{b}{sin(B)} \Leftrightarrow \tag{8}$$

$$a = \frac{b \cdot sin(A)}{sin(B)} \tag{9}$$

$$\text{Da B} = 90 \text{ og} sin(90) = 0 \text{ undlades nævneren} \tag{10}$$

$$a = b \cdot sin(A) \tag{11}$$

$$y_{TBe} = y_{Robot} + |RTB_e| \cdot sin(angleTotal) \tag{12}$$

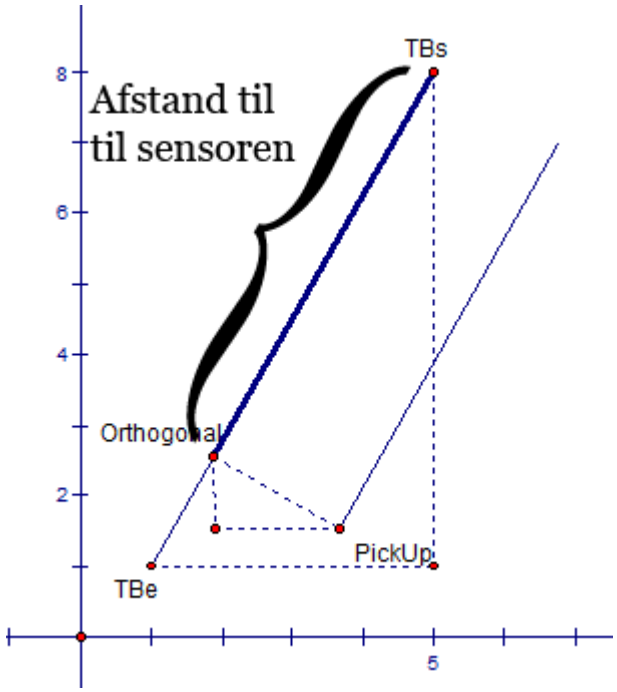
$$x_{TBe} = x_{Robot} + |RTB_e| \cdot sin\left((180 - 90 - angleTotal) \cdot \frac{\pi}{180}\right) \tag{13}$$

Her ganges der med $\frac{\pi}{180}$ for omskrivning fra grader til radianer.
De $180 - 90 - angleTotal$ er der sidste vinkel i trekanten.

Nu kan vinklen $\angle TBeRobotTBs$ findes ved cosinus-relationen (linje 19), hvorefter TBs's y-koordinat (linje 21) findes vha. $angleTotal$ (linje 20) og x-koordinat (linje 21) ligeledes.

Nu kan punkterne, hvor klodsen skal samles op beregnes:

Kodeudsnit 2: Funktionen 'CalculateSensorCoords()'



Figur 7: Illustration af beregning for SensorPickUp

```
1 internal void CalculateSensorCoords()
2 {
3     y4 = _TBe_y;
4     x4 = _TBe_x;
5     y3 = _TBs_y;
6     x3 = _TBs_x;
7
8     double a1;
9
10    a1 = (y4 - y3) / (x4 - x3);
11    if (double.IsNaN(a1))
12    {
13        a1 = 120;
14    }
15
16    double a2 = Math.Abs( -1 / a1);
17    double angleA1 = Math.Atan(a1);
18
19    double yOrtogonal;
20    if(angleA1 * deg < 0)
21        yOrtogonal = y3 + TBsSensor * Math.Sin(angleA1);
22    else
23        yOrtogonal = y3 - TBsSensor * Math.Sin(angleA1);
24
25    double xOrtogonal;
26
```

```
27      if(angleA1*deg < 0)
28          xOrtogonal = x3 + TBsSensor * Math.Sin((180 - 90 - ↩
                angleA1 * deg) * rad);
29      else
30          xOrtogonal = x3 - TBsSensor * Math.Sin((180 - 90 - ↩
                angleA1 * deg) * rad);
31
32      double GK = HalfBandSize * Math.Sin(Math.Atan(a2));
33
34      if(a1 > 0)
35          y5 = yOrtogonal - GK;
36      else
37          y5 = yOrtogonal + GK;
38
39      double GE = HalfBandSize * Math.Sin((180 - 90 - Math.Atan(a2)↩
            * deg) * rad);
40      x5 = xOrtogonal + GE;
41  }
```

De fundene punkter for transportbåndet læses ind i 4 lokale koordinater (linje 3-6).
Herefter beregnes hældningen på transportbåndet (linje 10) og det checkes om de skulle stå lige over hinanden (Ens x-koordinator). Hvis dette er tilfældet sættes hældningen til 120 (blot et højt tal) (linje 11-14).

Nu findes det punkt, hvor punktet *PickUp* står ortogonalt på (vinkelret):

$$a_2 = \frac{-1}{a_1} \tag{14}$$

Da denne skal bruges til at finde en længe, *skal* dette være positivt (linje 16).
Når hældningen, *a1*, er beregnet, findes den i grader (linje 17).

Først findes det ortogonale punkts y-koordinat:
Vha. Sinus-relationerne (ligning 11) findes forskellen på y-koordinatet.
Hvis *AngleA1* i grader er negativ lægges forskellen til *TBe*'s y-koordinat (linje 20-21).
Hvis *AngleA1* i grader er positiv trækkes forskellen fra *TBe*'s y-koordinat (linje 22-23).

Herefter findes det ortogonale punkts x-koordinatet på samme måde:
Ved at kombinere ligning 11, og reglen om, at en trekant er 180°, beregnes den sidste

vinkel i trekanten:

$$\text{Sidste vinkel} = 180° - 90° - angleA1 \tag{15}$$

Hvis *AngleR1* i grader er negativ lægges forskellen til *TBe*'s x-koordinat (linje 27-28).
Hvis *AngleA1* i grader er positiv trækkes forskellen fra *TBe*'s x-koordinat (linje 29-30).

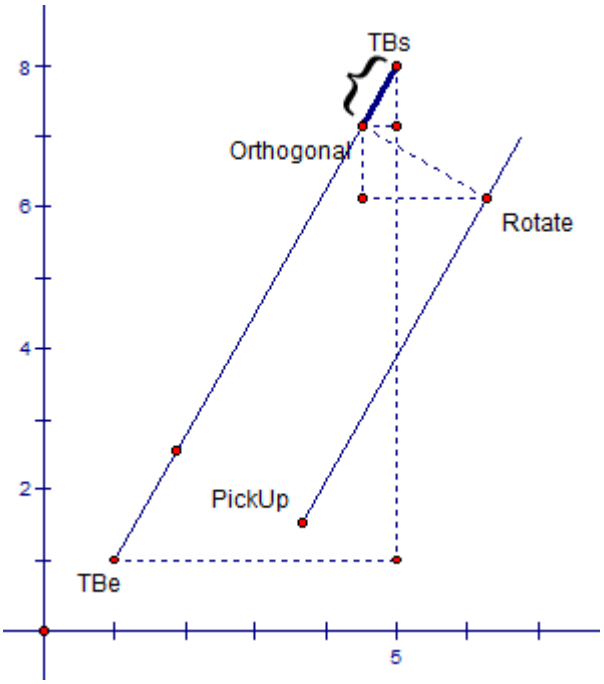
Når det ortogonale punkt er fundet, findes koordinatsættet til *PickUp* (angivet ved *x5* og *y5*):
Først findes forskellen på y-koordinatet (linje 32) med sinus-relationen. Hvis *a1* er positiv trækkes forskellen fra *yOrtogonal* (linje 34-35).
Hvis *a1* er negativ lægges forskellen til *yOrtogonal* (linje 34-35).

Herefter findes forskellen på x-koordinatet (linje 39) med sinus-relationen og formlen for den sidste vinkel (ligning 15).
Forskellen lægges til *xOrtogonal*.

Kodeudsnit 3: Udsnit af funktionen 'CalculateRotateCoords()'

```
19   double yOrtogonal;
20   if (angleA1 * deg < 0)
21       yOrtogonal = y3 + (TBsSensor - TBsPickUp) * Math.Sin(↩
             angleA1);
22   else
23       yOrtogonal = y3 - (TBsSensor - TBsPickUp) * Math.Sin(↩
             angleA1);
24
25   double xOrtogonal;
26
27   if(angleA1 * deg < 0)
28       xOrtogonal = x3 + (TBsSensor - TBsPickUp) * Math.Sin((180 -↩
            90 - angleA1 * deg) * rad);
29   else
30       xOrtogonal = x3 - (TBsSensor - TBsPickUp) * Math.Sin((180 -↩
            90 - angleA1 * deg) * rad);
```

På kodeudsnit 3 ses et udsnit af funktionen *CalculateRotateCoords()*. Det viste er det eneste der afviger fra funktionen *CalculateSensorCoords()*:



Figur 8: Illustration af CalculateRotateCoords()

Som illustreret på figur 8, beregnes længden til punktet *yOrtogonal*, hvor klodsen skal roteres, ved at trække afstanden fra sensoren til punktet fra længden til sensoren og gange med sinus-relationen (ligning 11).
Hvis vinklen (*angleA1)* er negativ, lægges *y3* (*TBs*'s y-koordinat) til beregningen (linje 21).
Hvis den er positiv trækkes det fra *y3* (linje 23).

xOrtogonal findes ved samme fremgangsmetode, hvor sinusrelationen (ligning 11) og ligning 15 kombineres.
Hvis vinklen (*angleA1)* er negativ, lægges beregningen til *y3* (*TBs*'s y-koordinat) (linje 30).
Hvis vinklen (*angleA1)* er positiv, trækkes beregningen fra *y3* (*TBs*'s y-koordinat) (linje 30).

For at bruge beregningerne kaldes funktionen *Messure(...*[1]*)*, hvorefter 2 arrays med index hver i sær på 2.
Herefter kaldes *PickUpCoords(out array1)* og *RotateCoords(out array2)* og de to koordinatsæt kan nu tilgås igennem de to arrays. Herefter

[1]Her mangler 3 parameter med forståelige navne