Berlin, Germany

Three dimensional coordinates into two dimensional coordinates transformation

Edward Gerhold

July 17, 2015

Version 0.2.9 (the is getting expanded version has a few new unshaped subsections)

Remark This text may contain miscounts and logical errors (the pseudo basis term is removed), *typos*, unclear phrases, remarks, during development. But this document is living and being edited. It is evolving from a transformation, to a documentation of the transformation plus a study of the spaces beetween which the transformation happens and how.

Contents

1 Introduction

On a piece of paper you see three coordinate axes pointing into three directions in space. In reality these vectors are two dimensional. Because they point into three directions on the paper, and not into the real space.

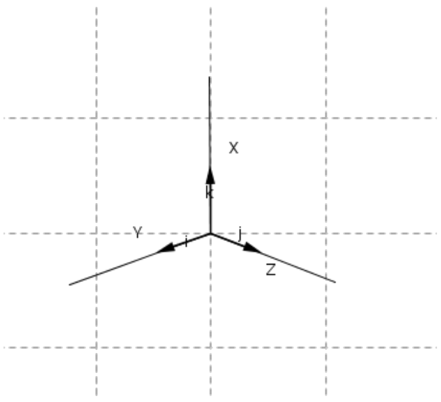

Figure 1: Picture of a right handed 3-D coordinate system with ijk-basis-vectors on the axes pointing into three dimensions. See [1] for introduction.

In this document we will design a R^{2x3} basis for the coordinate transformation. A basis is multiplied with the values of the coordinates to move for each component a piece, to end the move on the correct new point. In the case of cosines and sines, we move left and right and up and down, to tell you directly, what happens, when we multiply the coordinates with the matrix.

What we will do in the document

1. Choose angles for our coordinate axes around the unit circle to lay out three axes.
2. Write down the basis vectors for each coordinate axis
3. Assemble a matrix with the vector basis for a point by point transformation.

4. Read the example source code for a computer function, which is exactly two lines long. One for the new x and one for the new y.
5. Derive the generic case of transforming coordinate systems down to the plane.

2 Designing a R^{2x3} basis for our transformation from R^3 to R^2

2.1 The $_n$ index for $_{x,y,z}$ with $_x = 1$, $_y = 2$ and $_z = 3$

The index $_n$ in r_n, φ_n, $\vec{e}_n$ is the index for $_{x,y,z}$. For example φ_n stands $\varphi_x, \varphi_y, \varphi_z$. r_n stands for r_x, r_y and r_z. $\vec{e}_n$ is for $\vec{e}_x$, $\vec{e}_y$, $\vec{e}_z$ It is possible, that in the formulas x, y, z and $1, 2, 3$ may be used interchangeably. For example, when summing up the products of the coordinate components with the basis components, this happens. The formula is $\sum_{i=1}^{3} \vec{x}_i \vec{e}_i$, which is a sum of x, y, z and the $\cos\varphi_n$ terms in the first components of $\vec{e}_n$ for x' and a sum of x, y, z and the $\sin\varphi_n$ terms in the seconds components of $\vec{e}_n$ for y'.

2.2 φ_n the angles for the coordinate axes

Why do we need angles? May be the first question. My answer is, we will arrange the basis vectors, easily around a circle, by their angle. Since they are two dimensionsal. The circle is available in two dimensions. With the arrangement around a circle, we get the right numbers, same lengths, instead of guessing wild numbers.

First draw three axes of a 3-D coordinate system on a piece of paper. Draw the horizontal x-Axis through the origin of the drawn coordinate system. You could directly add the y-axis, to see a 2-D coordinate system carrying your two dimensional 3-D system. A system with three vectors pointing into three directions, originating in the origin of the real R^2*space*.

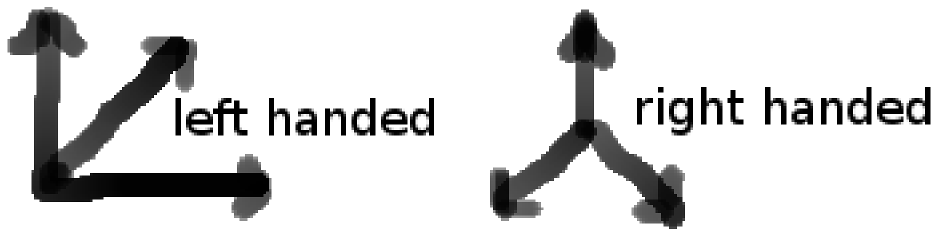

Figure 2: A left-handed and a right handed coordinate system. They just have different angles in our 2-D projection.

Each of the three vectors has an angle, counted from the horizontal positive x-axis, going counter-clockwise around the origin. The angle between the axes themselves isnt what we want. We want the angle beginning on the real 2-D x-axis, to feed the cos and sin functions with, when calculating the real numbers of each basis vector.

In this document, i will call the angles $\angle\varphi_n$ or just φ_n. If they are measured in degrees or radians depends on the cosine and sine functions you use. And on how you would like to read your own definition.

Let φ_n be the set of axis angles, one for each axis. I put them into a set in this document to simplify the access by using the index $_n$ together with $_{x,y,z}$ or $1, 2, 3$. φ_x or φ_1 is the angle of the x-axis. φ_y or φ_2 is the angle of the x-axis and φ_z or φ_3 is the angle of the x-axis.

$$\varphi_n := \{\varphi_x, \varphi_y, \varphi_z\} = \{\varphi_1, \varphi_2, \varphi_3\}$$

We will need the three angles for the axes shortly. So dont forget it over the next lines.

2.2.1 Degrees or radians?

Depending on the cosine and sine functions and the input value for the angles, you may have to convert the degrees to radians, or the other way round, the radians to degrees. For example, the JavaScript Math.cos and Math.sin functions take the values in radians.

example The function rad converts degrees to radians, its useful for computer functions taking radians.

$$\text{rad}(\phi) := \frac{\pi}{180} \times \phi, \phi \in R$$

Here is an example of three angles. The three axes have an angle of 120 degrees between each. But since we start counting counterclockwise and from the real horizontal axis of the plane, the angles are 210, 330, 90 in degrees, respectivly. And because of the cosine and sine functions taking radians, we convert the values to radians.

$$\varphi_x = \text{rad}(210), \varphi_y = \text{rad}(330), \varphi_z = \text{rad}(90)$$

$$\varphi_x = \frac{\pi}{180} \times 210 = \frac{7\pi}{6}, \varphi_y = \frac{\pi}{180} \times 330 = \frac{11\pi}{6}, \varphi_z = \frac{\pi}{180} \times 90 = \frac{\pi}{2}$$

example The function deg converts the other way round and from radians to degrees. You multiply your value with the reciprocal of PI/180, namely 180/PI and get the other way round.

$$\text{deg}(\phi) := \frac{180}{\pi} \times \phi, \phi \in R$$

If you would like to get hands on angles, cosines, sines, or need a refresher, [2] is a good choice. But also [1] and [4] bring unit circles, polar coordinates, sines, cosines and more informations.

2.3 r_n is the length of the unit on each axis

Before i show you the three vectors, and how to use them, we have to clear another piece of information belonging to each basis vector. In this document it is called r_n. The r is from radius. And it stands for the length of the basis vector. The length of the basis vector defines, how far a point in this direction will go by one unit.

The r originates from radius from the unit circle and from the parametrization of (x,y) via cosine and sine. In polar coordinates the cosine and sine are multiplied with r. Also to change the length of the hypotenuse, the vector $\vec{r}$, which is the third side to a triangle by cosine, sine and r. If r is left away, the length of the basis vector is 1. Or in other words, the distance $d((0,0),(x,y))$ from the origin to $(x,y) = (r\cos\varphi, r\sin\varphi)$ is 1, if $r = 1$ or if r is left away completely.

Pay attention to this point now, to keep the affine transformation, especially under rotation, correct. You should give all three axes the same r-value. I will define them in this document as r_n with one r_x, r_y and r_z for each coordinate axis, to keep it complete. But if you would like to change the units on your objects, i have to recommend, that you apply a local 3x3 basis with the disjoint unit lengths. This will keep the rotation correct. In the other case, the object would suddenly stretch the head, if you rotate it to the side, where the unit for example is longer.

So for the coordinate system, the best setting is $r_x = r_y = r_z$. Keeping them equal, you can rotate it realistic. But you dont need to keep the unit length of 1 for the vector. Elonginating the units on the axes make zooming transformations very easy.

2.3.1 Taking the norm of $\vec{e}_n$ to obtain r_n

If you have some existing basis and you would like to figure out, how long r is, you can go the other way round and take the norm of the vector. Taking the norm means to measure the length of the vector. This is done with the euclidean norm, or the 2-norm for regular purposes.

$r_n = \sqrt{\vec{e}_n \cdot \vec{e}_n} = \sqrt{(\vec{e}_n, \vec{e}_n)} = \left(\Sigma_{i=1}^{2} \vec{e}_i^2\right)^{\frac{1}{2}} = \|\vec{e}_n\|$
itt

With this formula you can not only measure the length of the basis vectors, but any vector in the R^3 and the R^2 space. More advanced measurements include the p-Norm, which is $\sqrt[p]{\sum_{i=1}^{n} |\vec{x}_i|^p} 1 \leq p \leq \infty$ and $\sup_{i=1..n} |\vec{x}_i|$ for $p = \infty$ and the max-Norm $\|\vec{x}\|_\infty = \sup\{|\vec{x}_i|\}$. There are matrix norms like $\|A\| = \max_{i=1..m} \sum_{j=1}^{n} A_{ij}$, which for example yields the largest row of a m by n matrix. Norms are used everywhere ing mathematics for measuring the lenghts of the vectors and matrices. And the distance function $d(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\|$ is used to measure the distance between to points or two vector tips. A vector space V with a distance function $d(x, y) = \|x - y\|$. is called a metric space (V, d). And a complete metric space with a norm, written $(V, \|\cdot\|)$, is a Banach space.

A vector can be normalized to give $\|\vec{x}\| = 1$, by dividing the vector components by the length, say $\vec{w}_{normalized} = \frac{\vec{w}}{\|\vec{w}\|}$. See the appendix for more on norms and for example for a proof of the normalization.

2.4 $\vec{e}_n$ are the three 2-D basis vectors

We have drawn some axes on a piece of paper and taken the angles starting from zero counterclockwise on the x-axis.

Now we will write down the three basis vectors. Each vector points from the origin exactly along the first unit of the together belonging axis.

Let $\vec{e}_n$ be the set of three two dimensional basis vectors. In this document and some literature and scripts, we call them $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$. Another well known names for the basis vectors are $\vec{i}$, $\vec{j}$ or $\vec{k}$ for example. That is equal to the picture of the coordinate axes at **??** in this document.

The three vectors point into the three directions of the three coordinate axes. Exactly along one unit, since we are going to define with them the length of one unit of the corresponding axis together with the positive direction of the coordinate axis. Multiplying the 2x3 basis with the 3x1 points later results in wonderful 2x1 points.

$$\vec{e}_n := \{\vec{e}_x, \vec{e}_y, \vec{e}_z\} = \{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$$

There we have the set for the three basis vectors. We give them the letter e and a subscript for the coordinate component in the numeric order of $x = 1, y = 2, z = 3$. To arrange these vectors we already got around the unit circle. To measure the angles, beginning on the horizontal coordinate axis or zero, until we reach the vector. The vectors point into the positive direction of the describing axis.

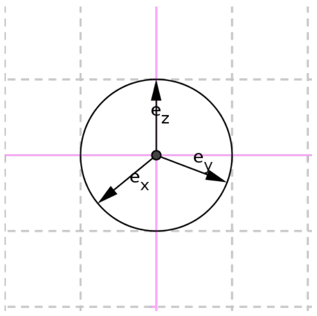

Figure 3: The three basis vectors point into the positive directions of the desired coordinate axis. They are arranged around a circle with the trigonometric functions of cosine and sine. The coordinate system shown is a righthanded coordinate system.

To reach all three (x,y) at the tips of the vectors, we will now pull out the cosine and sine functions and stuff them together with r and φ into a 2x1 vector with two components. So any (x,y) on one line from the origin to far distance can be reached like in polar coordinates[1] with the following parametriza-

[1]Interested readers may find in [1], [2] and [4] everything about polar coordinates, parametrization of x and y with cosine and sine, the unit circle and the distance or radius r and more to these topics.

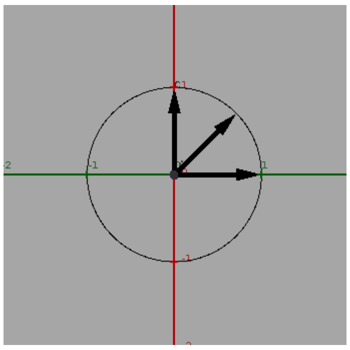

Figure 4: The three two dimensional basis vectors $\in R^{2x3}$ as a lefthanded coordinate system.

tion.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r\cos\varphi \\ r\sin\varphi \end{pmatrix}$$

Which can alternativly be written like $(x, y) = (r\cos\varphi, r\sin\varphi)$.

Modeling the three two dimensional basis vectors with this information, we get the following three two dimensional basis vectors. They point along the coordinate axes and are the ruler for our transformation.

$$\vec{e}_x := (r_x\cos(\varphi_x), r_x\sin(\varphi_x))^T = \begin{pmatrix} r_x\cos(\varphi_x) \\ r_x\sin(\varphi_x) \end{pmatrix}$$

$$\vec{e}_y := (r_y\cos(\varphi_y), r_y\sin(\varphi_y))^T = \begin{pmatrix} r_y\cos(\varphi_y) \\ r_y\sin(\varphi_y) \end{pmatrix}$$

$$\vec{e}_z := (r_z\cos(\varphi_z), r_z\sin(\varphi_z))^T = \begin{pmatrix} r_z\cos(\varphi_z) \\ r_z\sin(\varphi_z) \end{pmatrix}$$

Each component of (x,y,z) has now an own basis vector. By multiplying the cos terms for the x' and the sin terms for y' with the corresponding component of (x,y,z) and summing the three products up for each of x' and y', we directly obtain the right coordinate on the plane. All we would have to do is to connect the points again, or to fill the space between.

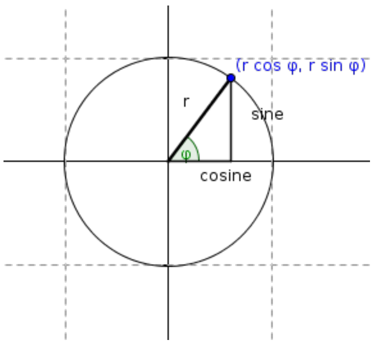

Figure 5: A picture of the unit circle, the hypotenuse r, the adjacent cosine, the opposite sine and the angle φ. It is a circle of radius r, and no longer the unit circle, if $r \neq 1$.

$$\boldsymbol{E}_{R^2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Figure 6: The standard basis for the R^2 spans up the two dimensional space. The basis is true in R^{2x3} but in R^2 just a linear combination of $\lambda \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mu \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

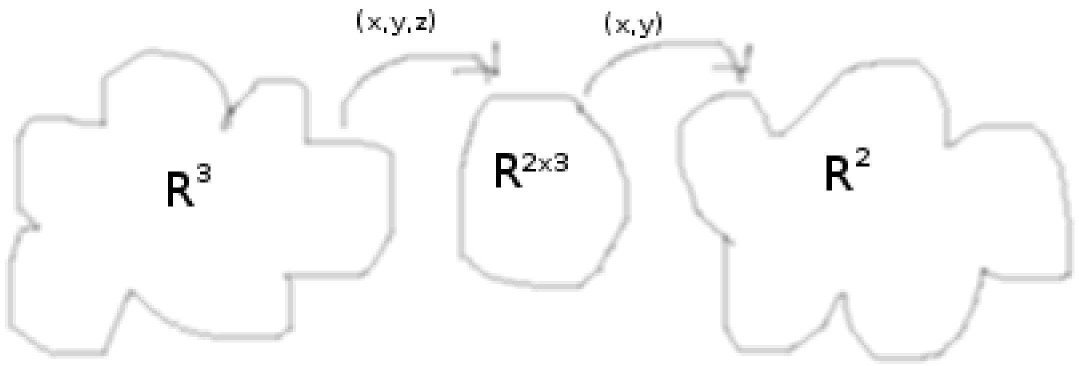

Figure 7: A temporary picture of the process. We multiply the 3-D points with the 2x3 matrix and get the 2-D points back.

2.5 About the vector basis lemma

2.5.1 The generic formula

I am talking about multiplying the coordinates with the new vector basis, which i state to be the same as the coordinate system we drew on a piece of paper at the beginning. We wrote down the angles, made out the unit length, and wrote down the three basis vectors with the information. Where is this coming from?

Every mathematics, physics or related course has a lesson, where the orthogonal basis of an objects coordinate system is introduced. A orthogonal basis is a set of 2 or three or up to infinite orthogonal or perpendicular vectors. They describe the coordinate system, the space, the dimensions, and one has to show for excercises, that the basis is linearly independent, that each basis vector points into its own dimension and not into the others.

The one lemma we need is this general theorem for multiplying a vector with the a basis of a target coordinate system.

The plane gives us two possible directions, to go horizontal or vertical. And in a cartesian coordinate system with infinite points, we can choose any direction around a center point (x,y). Which is in the case of our coordinate system the origin at (0,0,0) or (0,0). We will see later, that the zero vector stays in the origin fo both systems. Any not straight move will go horizontally or vertically by componentwise amounts. Any straight move horizontally or vertically will go by one of the components only.

Remark Edward was doing it right, then wrong (wrong with his pseudo-term). The basis is correct, but the space was R^{2x3}. In R^{2x3} the basis can be shown. In R2 it is just a linear combination of three scalars with three vectors. I will be able to formulate this out in the next versions and have nice images for explanation in mind, which i will draw for.

The point is, the general formula holds with a 2x3 basis.

The basis acts as a middler beetween the two spaces. We walk through the R^{2x3} when going from R^3 to R^2.

In R^2 the basis is linearly dependent, and not a basis, but multiplying the coordinates with, yield the right image of the set of points processed. On the other side, in R^{2x3} the basis can be explained, what i will do in the next versions. Have to read the text again, to catch my thread.

The formula for multiplying a vector with a base to get a new vector is this.[2]

$$\vec{w} = \sum_{i=1}^{n} \vec{v}_i \vec{e}_i$$

It is done componentwise for each row of the vector. n is the number of the source dimensions. In our case it is $n = 3$. We are summing three products for each component of the new vector. Our old $\vec{v}$ is a $\vec{v} \in R^3$.
With $\vec{v}_i$ as the coordinate component and $\vec{e}_i$ as the corresponding basis vector in the right component. $\vec{w}$ is the resulting new vector. The new vector $\vec{w}$ is a $\vec{w} \in R^2$.

In our scenario is $V \subseteq R^3, \vec{v} \in V$ and $W \subseteq R^2, \vec{w} \in W$.

2.5.2 Connection to ijk-Notation

This is also equal to

$$\vec{v} = x\vec{i} + y\vec{j} + z\vec{k}$$

what also explains, what the ijk-Notation means. If you dont use it already for determining determinants for calculating cross products (A.7). It is for describing a vector. Dont forget, our i, j, k basis is two dimensional, because we draw on a 2-D plane like the computer screen or a piece of paper.

With a 3x3 basis the vector $x\vec{i} + y\vec{j} + z\vec{k}$ is equal to $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$. But with a 2x3 basis the vector $x\vec{i} + y\vec{j} + z\vec{k}$ is becoming $\begin{pmatrix} x' \\ y' \end{pmatrix}$

2.6 Time to show the operation

The operation of multiplying the (x,y,z) coordinate with our R^{2x3} basis vectors in order is the following:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} xr_x \cos(\varphi_x) + yr_y \cos(\varphi_y) + zr_z \cos(\varphi_z) \\ xr_x \sin(\varphi_x) + yr_y \sin(\varphi_y) + zr_z \sin(\varphi_z) \end{pmatrix}$$

Right, this small formula brings over the R^{2x3} the unexpected images of the preimage from R^3 to R^2.

It is almost time to finish the matrix and to go through a set of points to draw the new set of resulting points. For this i close the explaining chapters and come to the part of the formal mathematical definitions.

3 My $R^3 \to R^2$ through R^{2x3} transformation theorem

Let V be the set of all points $(x, y, z) \in R^3$ which are about to become transformed. $V = \{\vec{v} = (x, y, z) | x, y, z \in R^3\}$. Let W be the set of all points $(x', y') \in R^2$ which are the result of the transformation $W = \{\vec{w} = (x', y') | x', y' \in R^2, (x', y') = \boldsymbol{A}\vec{v}\}$.

Remark study in progress.

[2]The formula can be found in many mathematics, chemistry and physics lecture scripts, and a good introduction is [3].

3.1 The transformation matrix

A $m \times n$ matrix is a rectangle or square of numbers.

$$\boldsymbol{A} = (a_{ij})_{i,j \in N} = \begin{pmatrix} a_{11} & ... & a_{mn} \\ \vdots & \ddots & \vdots \\ a_{m1} & ... & a_{mn} \end{pmatrix}$$

Matrix multiplication, from left to right in the matrix and from top to bottom in the vector, and that row by row, is achived by

$$\boldsymbol{A}\vec{v} = (\sum_{j=1}^{n} a_{ij}\vec{v}_j)_{i=1..m} = \begin{pmatrix} a_{11}v_1 + a_{12}v_2 + ... + a_{1n}v_n \\ \vdots \\ a_{m1}v_1 + a_{m2}v_2 + ... + a_{mn}v_n \end{pmatrix} = \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} = \vec{w}$$

This formula is not much different from the multiplication with a vector basis, but it accounts for rows and columns in the formula.While the vector basis multiplication implies the componentwise operations.

Definition 1 *Let $\boldsymbol{A}$ be the matrix containing the three, two dimensional and trigonometric, basis vectors in order, one each column. You get a rectangular 2x3 matrix $\boldsymbol{A} \in R^{2x3} : R^3 \to R^2$. With the basis vectors $\begin{pmatrix} r_n \cos \varphi_n \\ r_n \sin \varphi_n \end{pmatrix}$ in the three columns.*

$$\boldsymbol{A} := \begin{pmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \end{pmatrix} = \begin{pmatrix} r_x \cos(\varphi_x) & r_y \cos(\varphi_y) & r_z \cos(\varphi_z) \\ r_x \sin(\varphi_x) & r_y \sin(\varphi_y) & r_z \sin(\varphi_z) \end{pmatrix}$$

$\boldsymbol{A}$ should be treated as operator $\boldsymbol{A} \in R^{2x3} : R^3 \to R^2$. $(A)(\vec{x}) = \boldsymbol{A}\vec{x}$, $(\vec{x}) \mapsto \boldsymbol{A}\vec{x}$.

Remark. Make this another definition.

The operator lives itself in $O \subset R^{2x3}$. O for Operator is also $O \subset L(V, W)$, L(V,W) contains all linear operators from V to W. A is mapping the input, which is 3-D from $V \subseteq R^3$. onto the 2-D subset $W \subseteq R^2$.

Remark This chapter needs to be overworked.

3.2 The transformation

Theorem[The Fundamental Theorem of transforming 3-D Points into 2-D Points] 1 *If you multiply $\boldsymbol{A}$, the 2x3 matrix of the three two-dimensional basis vectors, with the three-coordinate point (x, y, z), the result is a two coordinate point, (x', y'). This point (x', y') is the correct point on the two dimensional plane, representing the point (x, y, z) from the three dimensional coordinate system, you are transforming.*

$$\boldsymbol{A} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

Applying the operator $\hat{\boldsymbol{A}}$ transforms the point $(x, y, z) \in R^3$ into a new point $(x', y') \in R^2$.
Proof:

$$\boldsymbol{A}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = x\vec{e}_x + y\vec{e}_y + z\vec{e}_z = \begin{pmatrix} xr_x\cos(\varphi_x) + yr_y\cos(\varphi_y) + zr_z\cos(\varphi_z) \\ xr_x\sin(\varphi_x) + yr_y\sin(\varphi_y) + zr_z\sin(\varphi_z) \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

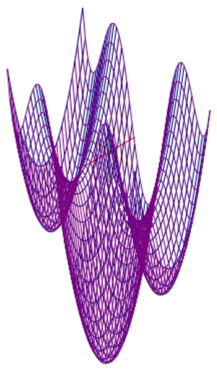

Figure 8: $f(x,y) = x^2 + y^2 + 3y\sin y$ from [-5,5] and [-3,3] on a Canvas2DRenderingContext

3.3 Computer implementations of the matrix and the transformation

3.3.1 Generic computer code

The following is example code for various computer systems.

```
x_ = x*r*cos(alpha) + y*r*cos(beta) + z*r*cos(gamma)
y_ = x*r*sin(alpha) + y*r*sin(beta) + z*r*sin(gamma)
```

3.3.2 JavaScript computer code

This is a full EcmaScript 6 snippet with all neccessary informations.

```
let rad = (deg) => Math.PI/180*deg;
let r_x = 1, r_y = 1, r_z = 1;
let phi_x = rad(220), phi_y = rad(330), phi_z = rad(90);
let xAxisCos = r_x*Math.cos(phi_x),
    yAxisCos = r_y*Math.cos(phi_y),
    zAxisCos = r_z*Math.cos(phi_z),
    xAxisSin = r_x*Math.sin(phi_x),
    yAxisSin = r_y*Math.sin(phi_y),
    zAxisSin = r_z*Math.sin(phi_z);
let transform2d = ([x,y,z]) => [
    x*xAxisCos+ y*yAxisCos+ z*zAxisCos,
    x*xAxisSin+ y*yAxisSin+ z*zAxisSin];
let transform2dAll = (P) => P.map(transform2d);

let examplePoints = transform2dAll([[1,2,3], [3,4,5], [14,24,15]]);
```

4 Corollaries

4.1 Converting four Dimensions down to two dimensions

The theorem can be used to handle more dimensions, for example can four two-dimensional vectors represent a 4-D space on the 2-D plane. They get converted into the correct 2-D points. For Example,

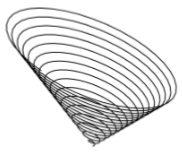

Figure 9: A conical helix (t/2*Math.cos(t), t*Math.sin(t), t) shown as (x,y,z)=f(t) with implement.html on a Canvas2DRenderingContext testing the javascript example code.

if you use a 2x4 matrix and convert all points at each instance of t you have a moving object into the direction of the fourth basis vector.

$$\boldsymbol{A} := \begin{pmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z & \vec{e}_t \end{pmatrix} = \begin{pmatrix} r_x\cos(\varphi_x) & r_y\cos(\varphi_y) & r_z\cos(\varphi_z) & r_t\cos(\varphi_t) \\ r_x\sin(\varphi_x) & r_y\sin(\varphi_y) & r_z\sin(\varphi_z) & r_t\sin(\varphi_t) \end{pmatrix}$$

Here the basis is four times of two dimensions. A 2x4 matrix with four two dimensional basis vectors, one for each axis.

$$\boldsymbol{A}\begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \sum_n \vec{e}_n\vec{x}_n = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

Proof:

$$\boldsymbol{A}\begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} xr_x\cos(\varphi_x) + yr_y\cos(\varphi_y) + zr_z\cos(\varphi_z) + zr_t\cos(\varphi_t) \\ xr_x\sin(\varphi_x) + yr_y\sin(\varphi_y) + zr_z\sin(\varphi_z) + zr_t\sin(\varphi_t) \end{pmatrix}$$

$$= x\vec{e}_x + y\vec{e}_y + z\vec{e}_z + t\vec{e}_t = \sum_n \vec{e}_n\vec{x}_n = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

The same method can be used, to convert points or vectors from any other number of dimensions, down to the xy-plane. It can also be used in a general m by n case.[3]

4.2 Alternative definition of the transformation by using dot products

Underways, i came to another conclusion. If i pull the two row vectors out of the matrix and define them as two column vectors, then i can dot each with the coordinate vector and write the dot product into the component of the resulting vector.

$$\vec{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \vec{c} = \begin{pmatrix} r_x\cos\varphi_x \\ r_y\cos\varphi_y \\ r_z\cos\varphi_z \end{pmatrix} \vec{s} = \begin{pmatrix} r_x\sin\varphi_x \\ r_y\sin\varphi_y \\ r_z\sin\varphi_z \end{pmatrix}$$

$$\vec{w} = \begin{pmatrix} \vec{v}\cdot\vec{c} \\ \vec{v}\cdot\vec{s} \end{pmatrix}$$

The result is $\vec{w} \in W$, $W \subseteq R^2$.

This can also be done with up to infinite dimensions which result in two coordinates then. Just add the dimensions to $\vec{v}, \vec{c}, \vec{s}$ and see.

[3]http://de.wikipedia.org/wiki/Abbildungsmatrix, shows the m by n case.

Proof:

$$\begin{pmatrix} \vec{v}\cdot\vec{c} \\ \vec{v}\cdot\vec{s} \end{pmatrix} = \begin{pmatrix} xr_x\cos(\varphi_x) + yr_y\cos(\varphi_y) + zr_z\cos(\varphi_z) \\ xr_x\sin(\varphi_x) + yr_y\sin(\varphi_y) + zr_z\sin(\varphi_z) \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

4.3 Easiest orthographic projection possible

Even less difficult to create a 3-D righthand coordinate system is this. Rotate the xy-plane by for example 225 degrees or $\frac{\pi}{180} \times 225 = \frac{15\pi}{12} = \frac{5}{4}\pi = 1.25\pi$ radians. Now the x-axis and the y-axis point downwards. This means, the real y-axis on the hardware is now free. So add the z-coordinate in by just adding it to y (Excercise, now try for the lefthand coordinate system yourself). This results in some acceptable orthographic projection.

$\angle$

$$\text{a} = 1.25\pi$$
$$\text{A} = \begin{pmatrix} \cos\angle a & -\sin\angle a \\ \sin\angle a & \cos\angle a \end{pmatrix}$$
$$\text{v} = (\text{x,y,z})^T, w = (v_1, v_2)^T = (x, y)^T$$
$$\boldsymbol{A}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$
$$\text{w} = \begin{pmatrix} x' \\ y' + z \end{pmatrix}$$

Example This is an EcmaScript 5 code. First rotate the xy plane to point downwards. Then add the z coordinate to y. That moves the point upwards again for the z-value. This results in an orthographic projection. Which is less flexible, but still one.

```
var angle = 1.25*Math.PI; // 225 deg
var cos = Math.cos(angle), sin = Math.sin(angle);
// for each point now
var u = x, w = y;
x = cos * u - sin * w; // rotate the point like we rotate the xy plane
y = sin * u + cos * w; // let the positive x,y axes point downwards
y += z; // the trick, let z now go upwards by just adding it to the vertical coord.
```

4.4 Orthogonality in the 2x3 matrix

Probably already found. 2-D (in each column, the two rows) mapping orthogonal, 3-D is linear (three moves left-right, three up-down) in the 2x3. In the 3x2 case the converse. Do never forget, that cos and sin are $\perp$ and form a right angle. TODO.

I mean. In each of the three column vectors, the map to the plane is defined by orthogonality of the two columns. Even if we change from the perpendicular sine and cosine to some weird numbers, it would be interpreted as x and y and depend on the base used in the r2.

5 Glossary

I am nativly a german speaking man. To reduce the risk of misunderstanding me, i will write down the terms, which i use in this document. So you can read from my definition, what i mean with and decide by yourself, whats the correct word, i wanted to use.

assumed english	german	personal definition
basis	Basis	A system of vectors, with which any coordinate has to be A basis transforms your coordinates into your desired coor A basis is regularly a set of linearly independent vectors. For example are $(1,0)$ and $(0,1)$ the basis for the R^2 and $(3,4)$ is a combination of $3 \times (1,0)$ and $4 \times (0,1)$,
vector, point, coordinate	Vektor, Punkt, Koordinate	A vector has a length and a direction. The tails of the vectors here meet in the origin. The point of the vector, or the word for the coordinates in the cartes The words point, vector, coordinate are used interchangab on what your numbers mean.
component	Komponente	A component of a vector or a matrix or a point is one of t

Remark incomplete.

6 Summary

6.1 Summary of all neccessary steps

1. Lay out the three basis vectors around a circle and write down the angles φ_n. Programmers have to write down a variable for anyways.
2. Write down the basis vectors $\vec{e}_n$ as $r_n \cos\varphi_n$ and $r_n \sin\varphi_n$ (two dimensional). Dont multiply with r_n for a unit length of 1 or multiply with r_n to change the length of the basis vector.
3. Put the three basis vectors $\vec{e}_n$ into a matrix A. Programmers can directly code the two lines of multiplication and forget the formal rest.
4. Iterate over your points and multiply each (x, y, z) with the matrix $\boldsymbol{A}$, which acts as a linear operator, and put (x', y') into your new set.[4]

Remark

About the word *unit*. I am not really sure, if i have to use *base vector* for a vector of any length and *unit vector* only for the *unit length* of 1. Because of the misleading mismatch with the *unit* of the thought *coordinate axes*, which the *base vector* defines, i tend in the first versions to misuse the word *unit vector* for both. If you find this, or any other formal mistake, be sure, it is not wanted :-) I will try to remove more of these spelling errors[5] in the next versions.

A Proving more rules of vector spaces

A.1 The origin stays in the origin

A trivial proof is to prove, that the zero vector $\vec{0} \in R^3$ maps to the zero vector $\vec{0} \in R^2$.

Proof:

$$\boldsymbol{A} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0+0+0 \\ 0+0+0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

A.2 Points along one axis

Another trivial proof is to prove, that coordinates lying on one axis are a multiple of the basis vector of the axis.

[4]Alternativly you can use the dot product to dot (x, y, z) with the cosine vector for x' and dot (x, y, z) with the sine vector for y'. The calculation is identical then.

[5]The *Gerholdian operator*, the *Gerholdian basis*, the *Gerhold projection matrix*, the *Gerhold transformation* are my favourite nicknames for my late discovery, making sure, the three two dimensional and trigonometric basis vectors, which i explained, sit in the matrix.

Proof:

$$\boldsymbol{A}\begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} ar_x \cos\varphi_x + 0 + 0 \\ ar_x \sin\varphi_x + 0 + 0 \end{pmatrix} = a\vec{e}_x$$

$$\boldsymbol{A}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 + r_y \cos\varphi_y + 0 \\ 0 + r_y \sin\varphi_y + 0 \end{pmatrix} = \vec{e}_y$$

$$\boldsymbol{A}\begin{pmatrix} 0 \\ 0 \\ -b \end{pmatrix} = \begin{pmatrix} 0 + 0 - br_z \cos\varphi_z \\ 0 + 0 - br_z \sin\varphi_z \end{pmatrix} = -b\vec{e}_z$$

A.3 Multiplications with constants

Another trivial proof is to show, that $\boldsymbol{A}(\lambda\vec{x}) = \lambda\boldsymbol{A}\vec{x}$. It doesnt matter, where you multiply with the constant. You can multiply the original vector, or the resulting vector. You reach the same point.

Proof:

$$\begin{aligned}
\boldsymbol{A}(\lambda\vec{x}) &= \boldsymbol{A}\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \end{pmatrix} \\
&= \begin{pmatrix} \lambda x r_x \cos(\varphi_x) + \lambda y r_y \cos(\varphi_y) + \lambda z r_z \cos(\varphi_z) \\ \lambda x r_x \sin(\varphi_x) + \lambda y r_y \sin(\varphi_y) + \lambda z r_z \sin(\varphi_z) \end{pmatrix} \\
&= \lambda \begin{pmatrix} x r_x \cos(\varphi_x) + y r_y \cos(\varphi_y) + z r_z \cos(\varphi_z) \\ x r_x \sin(\varphi_x) + y r_y \sin(\varphi_y) + z r_z \sin(\varphi_z) \end{pmatrix} \\
&= \lambda \begin{pmatrix} x' \\ y' \end{pmatrix} \\
&= \lambda\boldsymbol{A}\vec{x}
\end{aligned}$$

A.4 Additions and subtractions

Another trivial proof is to show, that $\boldsymbol{A}(\vec{v} + \vec{w}) = \boldsymbol{A}\vec{v} + \boldsymbol{A}\vec{w}$. It does not matter, if you add the original or the results . The outcome is the same point, the same vector.

Proof:

$$\begin{aligned}
\boldsymbol{A}\begin{pmatrix} x + u \\ y + v \\ z + w \end{pmatrix} &= \begin{pmatrix} (x+u) r_x \cos(\varphi_x) + (y+v) r_y \cos(\varphi_y) + (z+w) r_z \cos(\varphi_z) \\ (x+u) r_x \sin(\varphi_x) + (y+v) r_y \sin(\varphi_y) + (z+w) r_z \sin(\varphi_z) \end{pmatrix} \\
&= \begin{pmatrix} x r_x \cos(\varphi_x) + y r_y \cos(\varphi_y) + z r_z \cos(\varphi_z) \\ x r_x \sin(\varphi_x) + y r_y \sin(\varphi_y) + z r_z \sin(\varphi_z) \end{pmatrix} + \begin{pmatrix} u r_x \cos(\varphi_x) + v r_y \cos(\varphi_y) + w r_z \cos(\varphi_z) \\ u r_x \sin(\varphi_x) + v r_y \sin(\varphi_y) + w r_z \sin(\varphi_z) \end{pmatrix} \\
&= \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} u' \\ v' \end{pmatrix} \\
&= \boldsymbol{A}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \boldsymbol{A}\begin{pmatrix} u \\ v \\ w \end{pmatrix}
\end{aligned}$$

A.5 Rule of linearity

Corollary From the previous two proofs, it is obvious to see, that

$$\boldsymbol{A}(\lambda\vec{v}+\kappa\vec{w}) = \lambda\boldsymbol{A}\vec{v}+\kappa\boldsymbol{A}\vec{w} = \lambda\begin{pmatrix} x' \\ y' \end{pmatrix} + \kappa\begin{pmatrix} u' \\ v' \end{pmatrix}$$

which is a standard formulation of the rule of linearity. For example, you can find this rule in the form $\boldsymbol{A}(c\vec{x}+d\vec{y}) = c\boldsymbol{A}\vec{x}+d\boldsymbol{A}\vec{y}$ in [3], but also in every linear algebra 1 lecture script.

A.6 Dot product

The dot product, scalar product or inner product is the sum of the vector component products. $(\vec{v},\vec{w})$ is $\sum_{i=1}^{n}\vec{v}_i\vec{w}_i$.

$\sum_{i=1}^{n}\vec{v}_i\vec{w}_i = 0$ means, that $\vec{v}\perp\vec{w}$

The basis formula is this

$$(v,w) = \sum_{i=1}^{n}\vec{v}_i\vec{w}_i$$

A product with the zero vector.

$$(\vec{0},w) = \sum_{i=1}^{n}\vec{0}_i\vec{w}_i = 0$$

Linear combinations.

$$(\lambda\vec{v},\vec{w}) = \sum_{i=1}^{n}\lambda\vec{v}_i\vec{w}_i = \lambda\sum_{i=1}^{n}\vec{v}_i\vec{w}_i = \lambda(\vec{v},\vec{w})$$

$$(\lambda\vec{v},\vec{w}+\vec{x}) = \sum_{i=1}^{n}\lambda\vec{v}_i(\vec{w}_i+\vec{x}_i) = \lambda(\sum_{i=1}^{n}\vec{v}_i\vec{w}_i+\sum_{i=1}^{n}\vec{v}_i\vec{x}_i) = \lambda((\vec{v},\vec{w})+(\vec{v},\vec{x}))$$

$$(\lambda\vec{v},\kappa\vec{w}) = \sum_{i=1}^{n}\lambda\vec{v}_i\kappa\vec{w}_i = \lambda\kappa\sum_{i=1}^{n}\vec{v}_i\vec{w}_i = \lambda\kappa(\vec{v},\vec{w})$$

$$\begin{gathered}(\lambda\vec{v}+\mu\vec{x},\kappa\vec{w}+\nu\vec{y}) \\ = \textstyle\sum_{i=1}^{n}(\lambda\vec{v}_i+\mu\vec{x}_i)(\kappa\vec{w}_i+\nu\vec{y}_i) \\ = (\lambda\vec{v},\kappa\vec{w})+(\lambda\vec{v},\nu\vec{y})+(\kappa\vec{w},\mu\vec{x})+(\kappa\vec{w},\nu\vec{y}) \\ = \lambda(\kappa(\vec{v},\vec{w})+\nu(\vec{v},\vec{y}))+\kappa(\mu(\vec{w},\vec{x})+\nu(\vec{w},\vec{y}))\end{gathered}$$

A.7 The cross product

$$\begin{pmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{pmatrix} = \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}\vec{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix}\vec{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix}\vec{k} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

You write a new vector $x\vec{i}-y\vec{j}+z\vec{k}$ (pay attention to the minus) with the determinants, which you optain by scratching current column and the first row. You multiply the determinant with i, j, or k.

$$\vec{a}\times\vec{b} = (a_2b_3-a_3b_2)\vec{i}-(a_1b_3-a_3b_1)\vec{j}+(a_{1_b}2-a2b_1)\vec{k} = \vec{c}$$

If the cross product does not yield a new vector, but the zero vector, the two vectors are not on the same plane.

A.8 Normal vectors

Are perpendicular to the surface and give the orientation.

A.9 About the norm

The norm used is the euclidean norm, or the 2-norm. This is the square root of the sum of the squares of the absolute values of the components $\|\vec{x}\| = \sqrt{\sum_{i=1}^{n} |x_i|^2}$. In linear algebra, functional analysis and topology lectures there are three fundamental properties of the norm repeating. Definiteness, homogenity and the triangle inequality.

Definitness Show that $\|\vec{x}\| = 0 \iff \vec{x} = 0$

$$\|\vec{x}\| = \|\vec{0}\| = \sqrt{0^2 + 0^2} = 0$$

Homogenity Show that $\|a\vec{x}\| = |a|\|\vec{x}\|$

$$\|a\vec{x}\| = \sqrt{|a\vec{x}_1|^2 + |a\vec{x}_2|^2} = \sqrt{|a|^2(|\vec{x}_1|^2 + |\vec{x}_2|^2)} = |a|\sqrt{|\vec{x}_1|^2 + |\vec{x}_2|^2} = |a|\|\vec{x}\|$$

Triangle inequality Show that $\|\boldsymbol{A}(\vec{v} + \vec{w})\| \leq \|\boldsymbol{A}\vec{v}\| + \|\boldsymbol{A}\vec{w}\|$

This means, the path over two sides of the triangle is longer, than the side left over, no matter which way you turn. And it is a triangle, because the three points in the space are by at least one unit.

$$\sqrt{\sum_{i=1}^{n} |\vec{v}_i + \vec{w}_i|^2} \leq \sqrt{\sum_{i=1}^{n} |\vec{v}_i|^2} + \sqrt{\sum_{i=1}^{n} |\vec{w}_i|^2}$$

I tried to come closer to the proof of the triangle inequality. This was my first try.

$$\begin{aligned} &\sqrt{(a^2 + 2ab + b^2)} \leq \sqrt{(a^2)} + \sqrt{(b^2)} \\ &= \sqrt{(a+b)^2} \leq \sqrt{a^2} + \sqrt{b^2} \\ &= a + b \leq a + b \end{aligned}$$

In this case a+b are equal. But i could not get to the right proof.

$$(\sum_{i=1}^{n} |a_i + b_i|^2)^{\frac{1}{2}} \leq (\sum_{i=1}^{n} |a_i|^2)^{\frac{1}{2}} + (\sum_{i=1}^{n} |b_i|^2)^{\frac{1}{2}}$$

Older Remark. Here i could conclude at once, that each component row will be equal to the sum on the right side. But i can not.
At the end is it the fact, that the root pulled out of the larger ($(a+b)^2$ *is larger than* $a^2 + b^2$ *by* $2ab$) term is then a smaller number, than the two roots from the single letters being together, for example is $\sqrt{2} < \sqrt{1} + \sqrt{1}$. or 1.41 smaller than 1+1. The theorem seems to be established. But i did not prove.
My next attempt was rather logic than that. Pythagoras. I substituted the norm with a,b,c.

$$a = \|x - y\|, b = \|y - z\|, c = \|x - z\| |d(x,y) - d(y,z)| \leq d(x,z); |a - b| \leq \sqrt{a^2 + b^2} |a - b| \leq c$$

Remark. This wasnt what ive written. It shouldnt be, that i need my notes. But. Got to do it again.

$$\sqrt{a^2 - b^2} \leq \sqrt{a^2 + b^2}$$

Cauchy-Schwarz inequality There is another interesting inequality.
TODO

$$|\vec{v}\cdot\vec{w}|^2 \leq \|\vec{v}\|^2\|\vec{w}\|^2$$

The absolute value of the dot product squared is less or equal to the product of the two norms squared. Squaring the two norms to the right cancels the outer square root on the norms away, so that you multiply with the full vector length squared. So both sides are squares. Left is the dot product of the two vectors squared, on the right side the squared lengths of the two vectors are multiplied. So which square is larger? Ok, the formula is proof anyways and you know. But we have to get behind. And i for sure, too.

$$|\sum_{i=1}^{n}\vec{v}_i\vec{w}_i|^2 \leq ((\sum_{i=1}^{n}|\vec{v}_i|^2)^{\frac{1}{2}})^2((\sum_{i=1}^{n}|\vec{w}_i|^2)^{\frac{1}{2}})^2$$

$$|\sum_{i=1}^{n}\vec{v}_i\vec{w}_i|^2 \leq ((\sum_{j=1}^{n}\sum_{i=1}^{n}|\vec{v}_i|^2|\vec{w}_i|^2)^{\frac{1}{2}})^2 = |\sum_{i=1}^{n}\vec{v}_i\vec{w}_i|^2 \leq \sum_{j=1}^{n}\sum_{i=1}^{n}(|\vec{v}_i||\vec{w}_i|)^2$$

Remark This subsection is not complete. And may contain mistakes.

A.10 Normalizing a vector

If you wish to take the length of the vector, you take the norm $\|\vec{v}\|_V$ in three dimensions or $\|\vec{w}\|_W$ in two dimensions. If you wish to reduce or enlarge a vector to unit length, that $\|\vec{v}\| = 1$ or $\|\vec{w}\| = 1$, you can do this, by updating the or creating a new vector, by dividing the old vector components through the old vectors length. Taking the norm then, results in 1.

$$\vec{w}_{normalized} = \frac{\vec{w}}{\|\vec{w}\|} \implies \|\vec{w}_{normalized}\| = 1$$

Proof:

$$\vec{w} = \begin{pmatrix} a \\ b \end{pmatrix}$$

$$\|\begin{pmatrix} a \\ b \end{pmatrix}\| = \sqrt{a^2+b^2}$$

$$\vec{w}_{normalized} = \frac{\vec{w}}{\|\vec{w}\|} = \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2}} \\ \frac{b}{\sqrt{a^2+b^2}} \end{pmatrix}$$

$$\|\vec{w}_{normalized}\| = \sqrt{\left(\frac{a}{\sqrt{a^2+b^2}}\right)^2 + \left(\frac{b}{\sqrt{a^2+b^2}}\right)^2} = \sqrt{\frac{a^2+b^2}{a^2+b^2}} = \sqrt{1} = 1$$

A.11 Metrics

Where a norm is, there will be a metric induced. The measurement of the distance between two points is defined by the d-function. It is the length of the difference vector between the two points.

$$d(\vec{x},\vec{y}) = \|\vec{x}-\vec{y}\| = \sqrt{\sum_{i=1}^{n}|\vec{x}_i-\vec{y}_i|^2}$$

Metrics have three fundamental properties.

1. If the distance is zero, the vectors are equal.

$$d(x,y) = 0 \iff x = y$$

2. It does not matter, whether you read $d(x,y)$ or $d(y,x)$, the number must be equal.

$$d(x,y) = d(y,x)$$

3. The third one is the triangle inequality. Going over another point is always a step longer.

$$d(x,z) \leq d(x,y) + d(y,z)$$

Remark This subsection is not complete and has to be continued. The plan is to measure or estimate now the differences between the original coordinates and the new planar coordinates.

A.12 Transpose and TODO

A 2x3 matrix also has a transpose. Multiplying both result in two different square matrices. $\boldsymbol{A}^T\boldsymbol{A}$ is a 3x3 matrix. And $\boldsymbol{A}\boldsymbol{A}^T$ is a 2 by 2 matrix.

$$\begin{pmatrix} r_x\cos(\varphi_x) & r_y\cos(\varphi_y) & r_z\cos(\varphi_z) \\ r_x\sin(\varphi_x) & r_y\sin(\varphi_y) & r_z\sin(\varphi_z) \end{pmatrix}^T = \begin{pmatrix} r_x\cos(\varphi_x) & r_x\sin(\varphi_x) \\ r_y\cos(\varphi_y) & r_y\sin(\varphi_y) \\ r_z\cos(\varphi_z) & r_z\sin(\varphi_z) \end{pmatrix}$$

Multiplying out the transposes yield the following forms.

$\boldsymbol{A}\boldsymbol{A}^T$, a 2 by 2 matrix.

$$\boldsymbol{A}\boldsymbol{A}^T = \begin{pmatrix} \sum_{i=1}^3 r_n\cos\varphi_n & \sum_{i=1}^3 r_n^2\cos\varphi_n\sin\varphi_n \\ \sum_{i=1}^3 r_n^2\cos\varphi_n\sin\varphi_n & \sum_{i=1}^3 r_n\sin\varphi_n \end{pmatrix} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

In the 2x2 matrix $\boldsymbol{A}\boldsymbol{A}^T$ is $a_{ij} = a_{ji}$. Also in the 3x3 matrix $\boldsymbol{A}^T\boldsymbol{A}$ is $a_{ij} = a_{ji}$.
Remark I have (not) compared with the sin(A+B) formulas. The following terms are rewritable.

I will abbreviate $\cos\varphi_n$ with C_n and $\sin\varphi_n$ with S_n.

$\boldsymbol{A}^T\boldsymbol{A}$ a 3x3 matrix

$$\boldsymbol{A}^T\boldsymbol{A} = \begin{pmatrix} C_x^2 + S_x^2 & C_xC_y + S_xS_y & C_xC_z + S_xS_z \\ C_yC_x + S_yS_x & C_y^2 + S_y^2 & C_yC_z + S_yS_z \\ C_zC_x + S_zS_x & C_zC_y + S_zS_y & C_z^2 + S_z^2 \end{pmatrix} = \begin{pmatrix} r_x^2 & a & b \\ a & r_y^2 & c \\ b & c & r_z^2 \end{pmatrix}$$

Here i stopped the day. The page was full.

Remark Missing are $|\boldsymbol{A}\boldsymbol{A}^T|$ and $|\boldsymbol{A}^T\boldsymbol{A}|$ and $(\boldsymbol{A}\boldsymbol{A}^T)^{-1}$ and $(\boldsymbol{A}^T\boldsymbol{A})^{-1}$ and various tries to combine them to $P = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$.

TODO

A.13 The pseudo-inverse $\boldsymbol{A}^+ andleastsquares$

In a lecture script about numerical linear algebra i found the formulas for the pseudo inverses.

For a m by n matrix with m ¡ n it is $A^T(AA^T)^{-1}$

For a m by n matrix with m ¿ n it is $(A^TA)^{-1}A^T$

TODO

B An alternative graphics algorithm

Remark This section is new on July 10.

It is obvious, that we want to draw some graphics on our 2-D Canvas.

B.1 Origin

Setting the origin is an easy task. Assuming, the regular origin is at (0,0,0) and (0,0), we just need to add the shift to the coordinate. You can shift the 3-D Origin or the 2-D Origin.

```
x = Ox + x;
y = Oy + y;
z = Oz + z;
```

B.2 Translation

You simply add the translation vector to each point.

```
x = Tx + x;
y = Ty + y;
z = Tz + z;
```

B.3 Scaling

To scale the object you just have to multiply with the constant.

```
x = Sx * x;
y = Sy * y;
z = Sz * z;
```

B.4 Skewing

Skewing or shearing is not difficult. I took a skewing matrix and forgot about the empty fields.

```
u = x, v = y, w = z;
x =        u + k_xy*v + k_xz*w;
y = k_yx*u + v        + k_yz*w;
z = k_zx*u + k_zy*v + w;
```

B.5 Local 3x3 basis

If you wish to introduce different units for different directions, you have to apply a local basis, if the picture is moving angular. Applying the local 3x3 basis to an object makes sure, it will be rotatable, but without side effects. If you would change the units of r on the projection, then the rotation will give unrealistic results, since suddenly the object stretches to an unexpected size, when entering the zone.

The matrix applied locally is a 3x3 matrix $\begin{pmatrix} xBX & yBX & zBX \\ xBY & yBY & zBY \\ xBZ & yBZ & zBZ \end{pmatrix}$. For example is $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ is the original and orthonormal (orthogonal and unit length vectors) standard basis for the R^3 and the result is the same as if you do not apply any basis to the object, as the assumed default coordinate system in R^3 is orthonormal.

```
u = x, v = y, w = z;
x = u*xBX + v*yBX + w*zBX;
y = u*xBY + v*yBY + w*zBY;
z = u*xBZ + v*yBZ + w*zBZ;
```

This of course transforms the object by the directions and the length of the three three dimensional basis vectors.

B.5.1 Creating a 3x3 basis with cross products

```
function cross(a,b) {
    return [a[1]b[2]-a[2]b[1], -a[0]b[2]+a[2]b[0], a[0]b[1]-a[1]b[0]];
}
```

// if you look and remember A.7 you see the third determinant only giving (1-0)k

```
var u = [1,0,0];
var v = [0,1,0];
var w = cross(u,v);
// v = [0,0,1]
```

// if you look and remember A.7 you see the third determinant only giving (-1-0)k

```
var u = [-1,0,0];
var v = [0,1,0];
var w = cross(u,v);
// v = [0,0,-1]
```

B.6 Rotation

Rotating the object can be done in three dimensional space by applying the regular rotation matrices.

```
// once
    var rotxcos = Math.cos(xAngleRad), rotxsin = Math.sin(xAngleRad);
    var rotycos = Math.cos(yAngleRad), rotysin = Math.sin(yAngleRad);
    var rotzcos = Math.cos(zAngleRad), rotzsin = Math.sin(zAngleRad);
// for each point
    u = x, v = y, w = z;
    y = v * rotxcos - w * rotxsin
    z = v * rotxsin + w * rotxcos
    u = x, v = y, w = z;
    x =  u * rotycos + w * rotysin;
    z = -u * rotysin + w * rotycos;
    u = x, v = y, w = z;
    x = u * rotzcos - v * rotzsin;
    y = u * rotzsin + v * rotzcos;
```

B.7 Frustum and Perspective

Apply the perspective to the 3x3 set of points before projecting.

TODO

B.8 Dot product

The dot product or inner product or scalar product is the sum of the component products and one of the most important basic formulas in space.

```
function dot(a,b) {
    var sum = 0;
    for (var i = 0, j = a.length; i < j; i++) sum += a[i]*b[i];
    return sum;
}
```

B.9 Norm

The euclidean norm is the length of the vector. Its the square root pulled out of the sum of all components squared.

```
function norm(a) {
    var sum = 0;
    for (var i = 0, j = a.length; i < j; i++) sum += a[i]*a[i];
    return Math.sqrt(sum);
}
```

A p-Norm version, the second argument is the exponent p and p-th root.

```
function norm(a,p) {
    var sum = 0;
    if (p===undefined) p = 2;
    if (p===Infinity) {
        var max = 0;
        for (var i = 0, j = a.length; i < j; i++) {
            max = Math.max(Math.abs(a[i]), max);
        }
        return max;
    }
    for (var i = 0, j = a.length; i < j; i++) sum += Math.pow(Math.abs(a[i]),p);
    for (i = 2; i < p; i++) sum = Math.sqrt(sum);
    return sum;
}
```

B.10 Metric

The distance function gives us the distance beetween two points, that is the length of the vector from tip to tip.

```
function d(a,b) {
    var sum = 0;
    for (var i = 0, j = a.length; i < j; i++) sum += Math.pow(a[i]-b[i],2);
    return Math.sqrt(sum);
}
```

B.11 Code Example

Here is an implementation of these function together with the EcmaScript 6 snippet in modern EcmaScript 5.

```
(function (exports) {

function rad(deg) {
    return Math.PI/180*deg;
}

var r_x = 1, r_y = 1, r_z = 1;

var phi_x = rad(210), phi_y = rad(330), phi_z = rad(90);

var xAxisCos = r_x * Math.cos(phi_x),
    yAxisCos = r_y * Math.cos(phi_y),
    zAxisCos = r_z * Math.cos(phi_z),
    xAxisSin = r_x * Math.sin(phi_x),
    yAxisSin = r_y * Math.sin(phi_y),
    zAxisSin = r_z * Math.sin(phi_z);

function transform2d(vec3) {
    return [
    vec3[0]*xAxisCos + vec3[1]*yAxisCos + vec3[2]*zAxisCos,
    vec3[0]*xAxisSin + vec3[1]*yAxisSin + vec3[2]*zAxisSin
    ];
}

function transform2dAll(avec3) {
    return avec3.map(transform2d);
}

function settrans(op) {
    if (op.phi_{n}) {
    phi_x = op.phi_{n}[0];
    phi_y = op.phi_{n}[1];
    phi_z = op.phi_{n}[2];
    }
    if (op.r_{n}) {
    r_x = op.r_{n}[0];
    r_y = op.r_{n}[1];
    r_z = op.r_{n}[2];
    }
    xAxisCos = r_x * Math.cos(phi_x);
    yAxisCos = r_y * Math.cos(phi_y);
    zAxisCos = r_z * Math.cos(phi_z);
    xAxisSin = r_x * Math.sin(phi_x);
    yAxisSin = r_y * Math.sin(phi_y);
    zAxisSin = r_z * Math.sin(phi_z);
}

function gettrans() {
    return {
    phi_{n}: [phi_x, phi_y, phi_z],
    r_{n}: [r_x, r_y, r_z]
    };
}

function draw2dAll(ctx, points2, scale) {
    ctx.save();
    scale = scale || 1;
    var x = scale * points2[0][0], y = scale * points2[0][1];
    ctx.moveTo(x,-y);
    ctx.beginPath();
    for (var i = 0, j = points2.length; i < j; i++) {
    x = scale * points2[i][0], y = scale * points2[i][1];
    ctx.lineTo(x,-y);
    ctx.moveTo(x,-y);
    }
    ctx.closePath();
    ctx.stroke();
    ctx.restore();
}

function rotate3dAll(xAngleRad,yAngleRad,zAngleRad, points3) {
    var rotxcos = Math.cos(xAngleRad), rotxsin = Math.sin(xAngleRad);
    var rotycos = Math.cos(yAngleRad), rotysin = Math.sin(yAngleRad);
    var rotzcos = Math.cos(zAngleRad), rotzsin = Math.sin(zAngleRad);
    var p, x, y, z, u, v, w;
    for (var i = 0, j = points3.length; i < j; i++) {
        p = points3[i], x = p[0], y = p[1], z = p[2];
        u = x, v = y, w = z;
        y = v * rotxcos - w * rotxsin
        z = v * rotxsin + w * rotxcos
        u = x, v = y, w = z;
        x = u * rotycos + w * rotysin;
        z = -u * rotysin + w * rotycos;
        u = x, v = y, w = z;
        x = u * rotzcos - v * rotzsin;
        y = u * rotzsin + v * rotzcos;
        p[0]=x;
        p[1]=y;
        p[2]=z;
    }
}

function rotate2dAll(zAngle, points2) {
    var rotzcos = Math.cos(zAngleRad), rotzsin = Math.sin(zAngleRad);
    var p, x, y, u, v;
    for (var i = 0, j = points2.length; i < j; i++) {
        p = points2[i], x = p[0], y = p[1];
        u = x, v = y;
        x = u * rotzcos - v * rotzsin;
        y = u * rotzsin + v * rotzcos;
        p[0]=x;
        p[1]=y;
    }
}

function translate3dAll(transvec, points3) {
    var p, x, y, z;
    var Tx = transvec[0],
    Ty = transvec[1],
    Tz = transvec[2];
    for (var i = 0, j = points3.length; i < j; i++) {
        p = points3[i];
        p[0]+=Tx;
        p[1]+=Ty;
        p[2]+=Tz;
    }
}

function translate2dAll(transvec, points2) {
    var p;
    var Tx = transvec[0],
    Ty = transvec[1];
    for (var i = 0, j = points2.length; i < j; i++) {
        p = points2[i];
        p[0]+=Tx;
        p[1]+=Ty;
    }
}

function scale3dAll(scaleX, scaleY, scaleZ, points3) {
    var p;
    for (var i = 0, j = points3.length; i < j; i++) {
        p = points3[i];
        p[0]*=scaleX;
        p[1]*=scaleY;
        p[2]*=scaleZ;
    }
}

function scale2dAll(scaleX, scaleY, points2) {
    var p;
    for (var i = 0, j = points2.length; i < j; i++) {
        p = points2[i];
        p[0]*=scaleX;
        p[1]*=scaleY;
    }
}

exports.gettrans = gettrans;
exports.settrans = settrans;
exports.transform2d = transform2d;
exports.transform2dAll = transform2dAll;
exports.rotate3dAll = rotate3dAll;
exports.rotate2dAll = rotate2dAll;
exports.scale3dAll = scale3dAll;
exports.scale2dAll = scale2dAll;
exports.translate3dAll = translate3dAll;
exports.translate2dAll = translate2dAll;
exports.rad = rad;
exports.draw2dAll = draw2dAll;

}(typeof exports != "undefined" ? exports : this));
```

Last but not least here is a code snippet doing all the things together.

```
var n = points3.length;
var i = 0;
while (i < n) {
    var x,y,z, u,v,w;

    // local operations
    translate;
    rotate;
    scale;

    // world operations
    translate;
    rotate;
    scale;

}
```

References

Michael Corral, Schoolcraft College, Vector Calculus, GNU Free Documentation License, http://mecmath.net

Michael Corral, Schoolcraft College, Trigonometry, GNU Free Documentation License, http://mecmath.net

Gilbert Strang, MIT, Linear Algebra and its Applications. Fourth Edition.

Gilbert Strang, MIT, Calculus. MIT OpenCourseWare Supplemental Resources. http://ocw.mit.edu

Michael Corral, Schoolcraft College, Latex Mini Tutorial, http://mecmath.net

Manuela Jürgens, Thomas Feuerstack, Fernuniversität Hagen, LaTeX, eine Einführung und ein bisschen mehr..., a026_latex_einf.pdf

Dr.Jan Rudl, Technische Universität Dresden, Fachbereich Mathematik, Einführung in LaTeX, LaTeX-Kurs.pdf