Berlin, Germany

Three dimensional coordinates into two dimensional coordinates transformation

Edward Gerhold

August 5, 2015

Version 0.3.17

Remark. This is a development version. And has chaotic parts This file contains typos, unwanted logical mistakes and miscounts, and a maybe accidental letters, which came from a suddenly appearing double cursor in the editor. This is my first LaTeXbut not my last. If you ever print this, print 4 on 1. This saves paper. This document is unfinished and not completly shaped.

I think i have found the first words for my definition of R2x3 in the first of the sections about.

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.

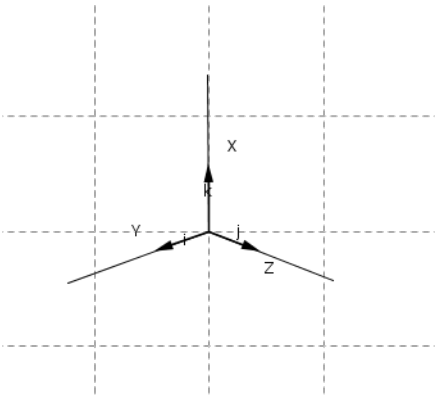

Figure 1: Picture of a right handed 3-D coordinate system with ijk-basis-vectors on the axes pointing into three dimensions. See (1) for introduction.

In this document we will design a $\mathbb{R}^{2\times 3}$ 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. Read other versions of the transformation, with functions, for example.
6. Derive the generic case of transforming coordinate systems down to the plane.

2 Designing a $\mathbb{R}^{2\times3}$ coordinate system for our transformation from $\mathbb{R}^3$ to $\mathbb{R}^2$

2.1 Axis vector

The words basis, axis vectors, coordinate system are used interchangably. And from those choices we talk about a coordinate system.

2.2 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'.

Being on point explaining indices, i should also explain this. The coordinates x, y, z in the vector $\vec{v}$ are the same as the components $\vec{v}_1, \vec{v}_2, \vec{v}_3$. And the components x', y' in $\vec{w}$ are equal to $\vec{w}_1, \vec{w}_2$.

2.3 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*.

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.

We will arrange the three axes for x, y and z around the circle by choosing angles.

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.

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

The angles are going around a circle, so they are limited by a modulus operation internally.

$$\varphi_n \in [0, 2\pi] \text{ in radians}, \varphi_n \in [0, 360] \text{ in degrees}, \varphi_n \in \mathbb{R}$$

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

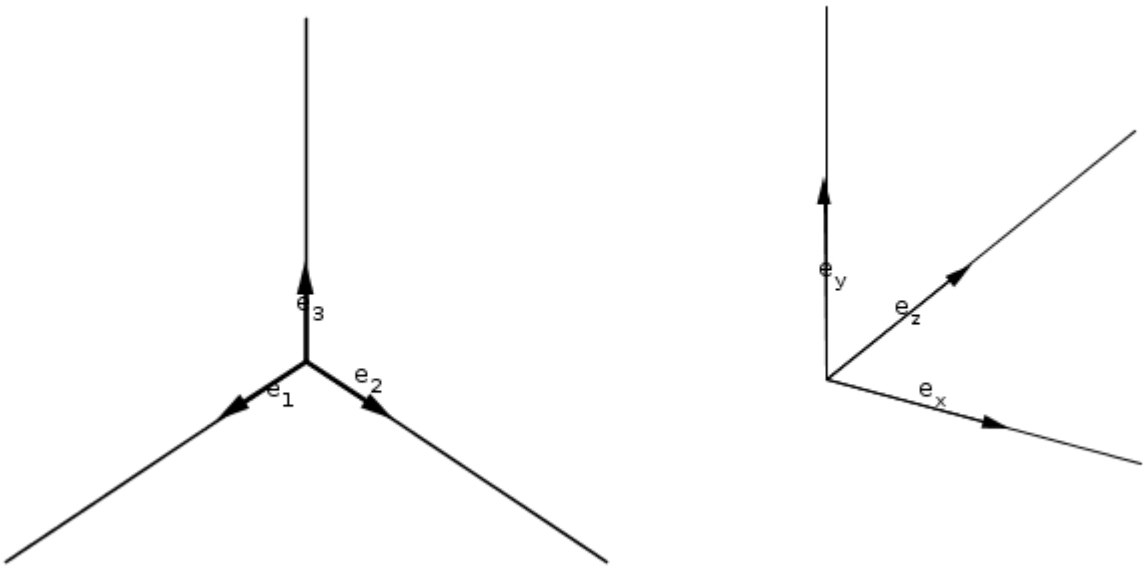

Figure 2: A right handed (z-up) and a left handed coordinate system. They just have different angles in our 2-D projection.

2.3.1 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 1 The function rad converts degrees to radians, its useful for computer functions taking radians.

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

Example 2 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 3 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.

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

Example 4 The first example was for the righthand coordinate system. Here are some angles for a lefthand system.

$$\varphi_x = \frac{\pi}{180} \times 0 = 0, \varphi_y = \frac{\pi}{180} \times 90 = \frac{\pi}{2}, \varphi_z = \frac{\pi}{180} \times 45 = \frac{\pi}{4}$$

If you would like to get hands on angles, cosines, sines, or need a refresher, (2) is a good choice. And as well (1) and (4) teach unit circles, polar coordinates, sines, cosines and wonderful mathematics.

2.4 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.

$$r_n := \{r_x, r_y, r_z\} = \{r_1, r_2, r_3\}$$

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.

The number r is for the length of one unit on each axis.

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.

2.5 $\vec{e}_n$ 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 described axis.

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.

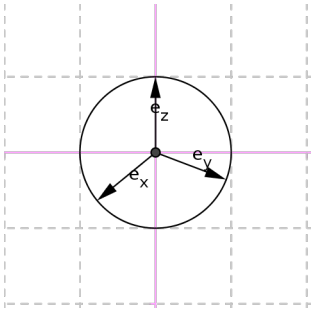

Figure 3: The three basis vectors point into the positive directions of the desired coordinate axes each. They are arranged around a circle with the trigonometric functions of cosine and sine. The coordinate system shown is a righthanded coordinate system.

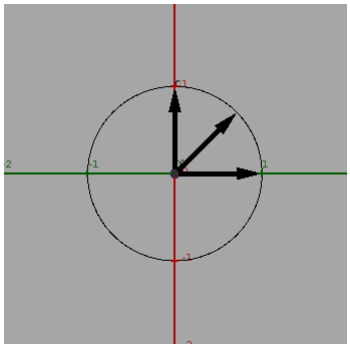

Figure 4: The three two dimensional basis vectors $\in \mathbb{R}^{2\times3}$ as a lefthanded coordinate system.

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.

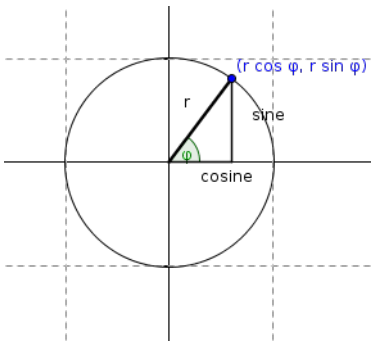

Figure 5: A picture of the unit circle, the hypotenuse r, the adjacent cosine, the opposite sine and the angle φ. It is a circle of radius r, and no longer the unit circle, if $r \neq 1$.

2.6 Now we need the vector basis theorem

2.6.1 The general formula for importing a vector into a coordinate system

Last section 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 object it's coordinate system is introduced. Orthogonality has some wonderful properties, and solving differential equations and other complicated systems take help from orthogonal vector sets.

A orthogonal basis is a set of 2 or three or up to infinite orthogonal (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.

Another excercise is to orthogonalize the existing vectors with Gram-Schmidt.

We want to design a coordinate system with three coordinates and two dimensions. At least one vector has to be linearly dependent of both basis vectors. We want to design a basis, or better a linear mapping, or best a coordinate system, which is a mix of both dimensions. We will use cosine and sine. We combine the three coordinates for three proportional horizontal moves. We combine the three coordinates for three vertical moves. Proportional to the coordinates and possibly with positive or negative amounts (up or down with sine, right or left with cosine) depending on the direction of the coordinate axis.

Remark. My article broke in when first time touching the linear dependence after being sure this formula works and "i have got some basis, right?". But i think, we are making progress. Now let us continue designing the axis vectors. We will look at the formula now.

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

The first time i had the idea, it was, "now try multiplying the coordinates with a basis." "But hey, they must be 2-D."

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.

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

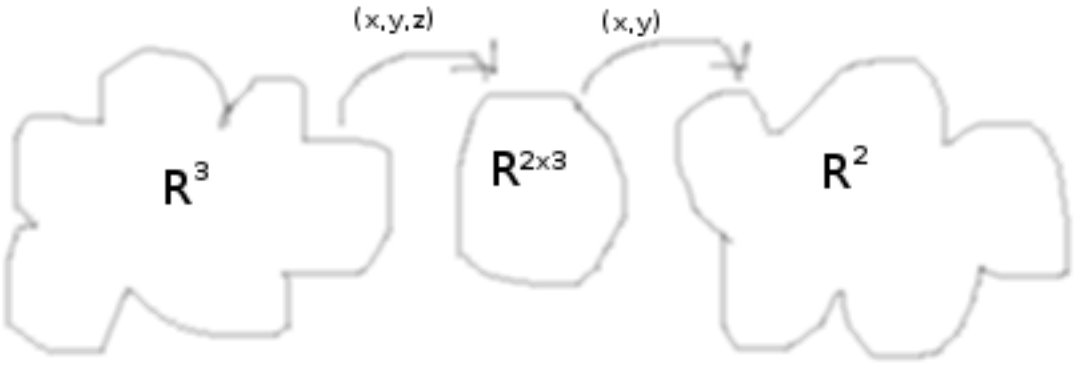

Figure 6: A temporary picture of the process. We multiply the 3-D points with the 2x3 matrix and get the 2-D points back.

$$\boldsymbol{E}_{\mathbb{R}^2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \boldsymbol{E}_{\mathbb{R}^3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{E}_{\mathbb{R}^{2\times3}} = \begin{pmatrix} r_x \cos\varphi_x & r_y \cos\varphi_y & r_z \cos\varphi_z \\ r_x \sin\varphi_x & r_y \sin\varphi_y & r_z \sin\varphi_z \end{pmatrix}$$

Figure 7: The standard basis for the $\mathbb{R}^2$ spans up the two dimensional space. When the three coordinates, which were a linear combination of $\lambda \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \mu \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \nu \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ are combined into two coordinates, they become a linear combination of $\lambda \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mu \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. For sure, λ is the sum of the cosine terms with the coordinates and μ is the sum of the sine terms with the coordinates in the two dimensions.

By taking 2-D vectors for three coordinates, we map directly onto the plane.

The formula for multiplying a vector with a basis 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 $\vec{v} \in \mathbb{R}^3$.
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 $\vec{w} \in \mathbb{R}^2$.

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

2.6.2 Connection to ijk-Notation

This is also equal to

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

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

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 $\begin{pmatrix} x' \\ y' \end{pmatrix}$

2.6.3 Coordinate system

2.7 Time to show the operation

The operation of multiplying the (x,y,z) coordinate with our $\mathbb{R}^{2\times3}$ coordinate axis 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 $\mathbb{R}^{2\times3}$ the unexpected images of the preimage from R^3 to R^2.

Remark. Meanwhile i am ready to say $\mathbb{R}^{2\times3}$ image, and to believe, that this coordinate system is spanning the $\mathbb{R}^{2\times3}$ up (spread into the three dimensions) on the plane.

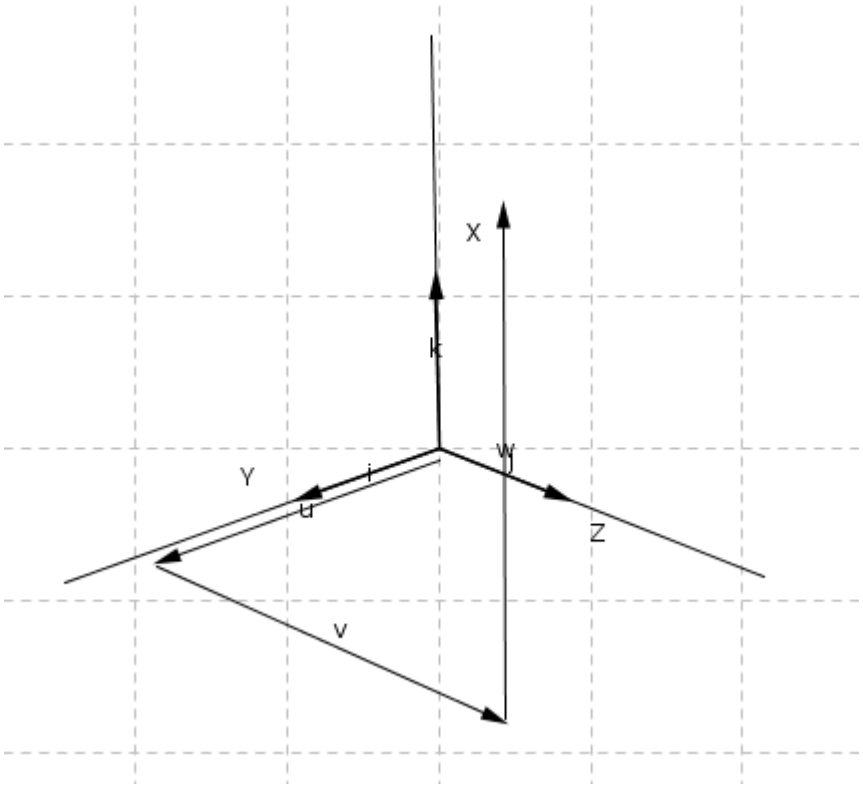

Figure 8: The path a point goes from the origin. Along the first axis, then from that parallel to the second along that axis, and last parallel to the third axis as many units as the coordinate says. You can not see on this picture, how it is deconstructed by cosine and sine into left and right moves. To see, just draw the two missing sides of the triangles under each move. The z axis has a cosine of 0. I will paint a new picture for.

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. (Were i will find alternatives for the matrix and related rules and laws, helpful for the understanding of the happenings.)

Remark about the document structure. LaTeXand i are new to each other. For the theorems, proofs, defintions, corollaries, examples there is the possibility of a personal layout, which i have not prepared yet. And additionally, the following will contain some things, where real mathematicians would start to smile. But i will do my best to correct any of my passages over the next time until i reach v1.0.0.

3 The transformation $\mathbb{R}^3 \to \mathbb{R}^2$ yields a $\mathbb{R}^{2\times3}$ image

3.1 Defining the Topology on the

Remark. This subsection is started on July 31.

Let V be an open set in $\mathbb{R}^3$.

Let $B(\vec{v},\epsilon)_3$ be a standard environment in the 3-space.

Let W be an open set in $\mathbb{R}^2$.

Let $B(\vec{v},\epsilon)_2$ be a standard environment in the 2-space.

Let the euclidean norm $\|\cdot\|_2$ be the default norm for three and two dimensions.

Let the $d(x,y)_2 = \|x-y\|$ be the according metric.

Let $f : V \to W$ be a continuous functional, and $A : V \to W$ be a rectangular matrix. Both map with equal operations V to W.

Let V be the set of all points $(x,y,z) \in V \subset R^3$ which are about to become transformed. $V := \{\vec{v} = (x,y,z)^T | x,y,z \in \mathbb{R}, \vec{v} \in V \subset \mathbb{R}^3\}$.

Let W be the set of all points $(x',y') \in W \subset R^2$ which are the result of the transformation $W; = \{\vec{w} = (x',y')^T | \vec{w} \in W \subset \mathbb{R}^2 x', y' \in \mathbb{R}, \boldsymbol{A}\vec{v} = (x',y')^T\}$.

Remark. The topology has still to be defined extensivly in this document.

3.2 Matrix version

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

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

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

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

This formula is not much different from the multiplication with a vector basis, but it also accounts for the rows in the formula. The vector basis multiplication implies the componentwise row operations by using vectors.

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 $\boldsymbol{A} \in \mathbb{R}^{2\times 3} : \mathbb{R}^3 \to \mathbb{R}^2$. With the coordinate axis vectors $\begin{pmatrix} r_n \cos\varphi_n \\ r_n \sin\varphi_n \end{pmatrix}$ in the three columns.*

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

Remark. I commented the "linear map operator" definition out (linear map and operator is correct), because i have to write it again.

Proposition. My fundamental theorem of transforming 3-D Points into 2-D Points (Matrix) 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 $\boldsymbol{A}$ transforms the point $(x, y, z) \in V \subset \mathbb{R}^3$ into a new point $(x', y') \in W \subset \mathbb{R}^2$.
***Proof**:*

$$\boldsymbol{A}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = (\sum_{j=1}^{3} a_{ij}\vec{v}_j)_{i=1,2} = \begin{pmatrix} xr_x\cos(\varphi_x) + yr_y\cos(\varphi_y) + zr_z\cos(\varphi_z) \\ xr_x\sin(\varphi_x) + yr_y\sin(\varphi_y) + zr_z\sin(\varphi_z) \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

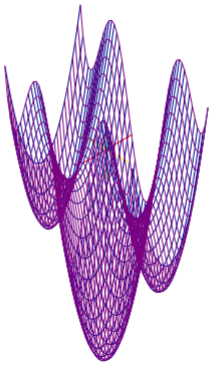

Figure 9: $f(x, y) = x^2 + y^2 + 3y\sin y$ from [-5,5] and [-3,3] on a Canvas2DRenderingContext

3.3 Vectorbasis version

3.3.1 Hamelbasis with broken law of linear independence (or just a linear mapping)

The theorem from Hamel says, that every vector space has a basis. And he gives a formula for this. The new vector in the new coordinate system is the sum of the coordinates multiplied with the basis vectors.

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

Remark. Have to fetch the Hamel-Theorem.

A Hamelbasis requires a linearly independent set of basis vectors. Which we can not provide. We change the dimension. The mapping yields the right image. So i will say, it is o.k. to break the rule of linear independence. This coordinate system is a special case.

Proposition. My fundamental theorem of transforming 3-D points into 2-D points (Vectorbasis) 1 *If you multiply the three linear dependent two dimensional vectors with the three dimensional coordinates, they are mapped correctly onto the two dimensional coordinate system.*

The operation is equal, but instead of working with three components in the new vector, we work with two components in the new vector. For each component we build the sum of the basis component multiplied by the belonging to coordinate, like we would do in the original form.

$$\vec{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\vec{w} = x\vec{e}_x + y\vec{e}_y + z\vec{e}_z$$

$$\sum_{i=1}^{n} \vec{v}_i \vec{e}_i = \begin{pmatrix} xr_x \cos(\varphi_x) + yr_y \cos(\varphi_y) + zr_z \cos(\varphi_z) \\ xr_x \sin(\varphi_x) + yr_y \sin(\varphi_y) + zr_z \sin(\varphi_z) \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} = \vec{w}$$

The difference is that we have a dimension less than coordinates, and with that at least one axis, that must be a mix of the other two. By default our image in our coordinate system is a linear combination of $\lambda \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \mu \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, but before that, we map with a linear function from three dimensions to two dimensions, by using our coordinate system as or like a basis.

Proposition. Breaking the rule of linear independence to map from 3-D to 2-D 1 *To transfer the points from 3-D to 2-D with the same formula, as mapping ordinary points with a vector basis into the corresponding coordinate system, it is o.k., to remove the third dimension from the basis and to multiply the 3-D coordinates with three 2-D vectors to get a correct mapping onto the projection plane.*

3.3.2 ijk-Notation Version

The ijk-Notation is well known from describing vectors, from calculating cross products over determinants, from coordinate systems showing the normalized ijk vectors along the axes. The formula is this

$$\vec{w} = x\vec{e}_x + y\vec{e}_y + z\vec{e}_z$$

This is a very natural way. This is a real sum. The coordinates x,y,z multiply each a basis vector. This is the ordinary constant or scalar multiplication. Then the three scaled vectors are summed up together. This gives us a new vector, the sum of the three vectors. I have explained this already earlier, you can use this notation for this purpose, now it is time to repeat it. For a picture, look at figures 1 and 2.3.

Proposition. The fundamental theorem of transforming 3-D points into 2-D points (ijk-Notation) 1 *If you write the vector down in ijk-Notation using the three two dimensional axis vectors, instead of three three dimensional linear independent basis vectors, the sum of the products with ijk and the coordinates, which is a new vector, equals the right vector on the 2-D plane.*

Proof:

$$x\vec{e}_x + y\vec{e}_y + z\vec{e}_z = x\begin{pmatrix} r_x\cos\varphi_x \\ r_x\sin\varphi_x \end{pmatrix} + y\begin{pmatrix} r_y\cos\varphi_y \\ r_y\sin\varphi_y \end{pmatrix} + z\begin{pmatrix} r_z\cos\varphi_z \\ r_z\sin\varphi_z \end{pmatrix} = \sum_{i=1}^{3} \vec{e}_i\vec{v}_i = \vec{w} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

3.4 Function version

The first days, i could not see the forest, because of all these trees. The operation can be written as function, or as part of a composition of functions. The big thing for this point by point transformation is the easy usability. For example, to create surface plots and other functional graphs from three dimensional space on a flat screen or printed paper.

3.4.1 The linear functional $f(\vec{x})$

We begin with $f(\vec{x}) : V \subset \mathbb{R}^3 \to\subset \mathbb{R}^2$.

$f(\vec{x})$ is mapping the three dimensional coordinates onto our designed coordinate system. The multiplication of the components with the horizontal and vertical displacements which are represented by the axes of our coordinate system is the fix content of our function. Assume we have the angles and units designed and the function is well defined for its purpose.

$$f(\vec{x}) := \begin{pmatrix} \vec{x}_1 r_x\cos\varphi_x + \vec{x}_2 r_y\cos\varphi_y + \vec{x}_3 r_z\cos\varphi_z \\ \vec{x}_1 r_x\sin\varphi_x + \vec{x}_2 r_y\sin\varphi_y + \vec{x}_3 r_z\sin\varphi_z \end{pmatrix}$$

Proposition. My fundamental theorem of transforming 3-D points into 2-D points (Functional)) 1 *The linear functional $\vec{f}(\vec{x})$ maps the points correctly from 3-D to 2-D. It is continuous in every point, the zero vector maps onto the zero vector. Passing a vector with three coordinates to the function results in a vector with two coordinates, which are the right coordinates on the 2-D screen.*

3.4.2 Composition of the functions $f \circ g$

There are various possibilities to combine the output of g and the input of f. The following functions are compositions of two functions and take some input and return our 2-D points. f is transformingthe vector returned by g. g is taking the input in all examples and f is reworking the coordinates. In other words, f is the same function as previously shown in 3.4.1.

Example 1. A call to $g(t)$ is returning a vector $\vec{v}$ passed to $f(\vec{x})$ by using the composition $f \circ g$: $f(g(t)) = \vec{w}$

$$g(t) := \begin{pmatrix} t\cos t \\ t\sin t \\ t \end{pmatrix}$$

Example 2. $g(x, y) = (x, y, z)$ this function will give us a surface plot. See figure 3.4.2.

$$g(x, y) := \begin{pmatrix} x \\ y \\ e^{-x^2-y^2} \end{pmatrix}$$

Example 3. A $g(x, y, z)$ or $g(\vec{x})$ a three-d or vector-valued function returning a three-d vector.

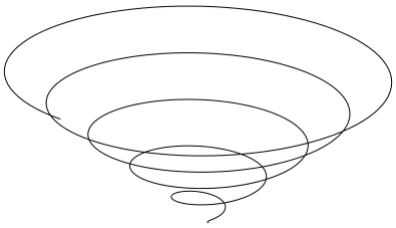

Figure 10: This is $g(t)$ from 3.4.2 in implement.html where implement.js from the repository is used once.

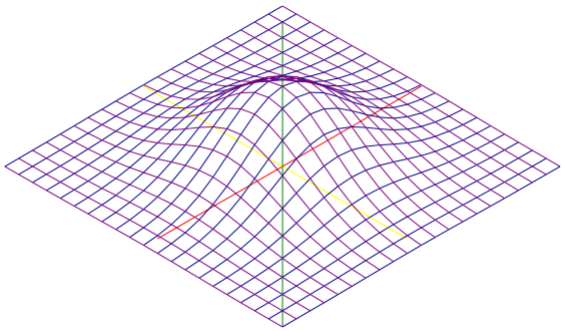

Figure 11: This is $\exp -x^2 - y2$ from 3.4.2 plotted with the cheap3danimate.html code within $[-2, 2] \times [-2, 2]$.

$$g(x, y, z) := \begin{pmatrix} x+1 \\ y \\ z-1 \end{pmatrix}$$

The vector field of this formula is shown in figure 3.4.2.
Remark. The vector field demo is primitive at this point.

$$g(\vec{x}) : \mathbb{R}^3 \to \mathbb{R}^3$$
$$f(\vec{x}) : \mathbb{R}^3 \to \mathbb{R}^2$$

Remark. This section is not finished. Not only the plot for the vector field, some sophisticated demo with a physics formula, but the compositions are themselves not explained. Additionally in the section about differentiation, the compositions have to be veryfied.

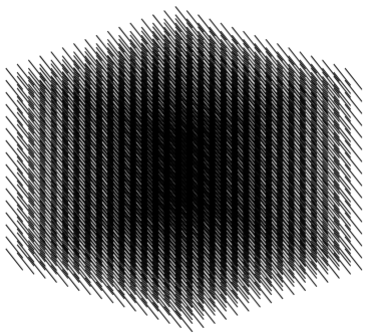

Figure 12: A 3-D vector field of a cubic section, this time of some random formula.

3.5 Computer implementations of the transformation

3.5.1 Generic computer code

One of the *main goals of this document* is to show the simplicity of transforming 3-D points into 2-D points for the use in small computer applications. For example for hand written small web applications. Say, you just want to draw a graph, 3-D on the 2-D Canvas and do not have the need for WebGL, or it is not available on your old target systems, which was the reason for me, to try it myself anyways.

This should be in a border box.

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)
```

These are the one and only two lines of code you need.

Only two lines of code needed to go from 3-D to 2-D points. (Computer Version) 1 *The only two lines of code you need to convert the coordinates on the computer. The new x value is summed up by multiplying each coordinate with the cosine term of the related axis vector. The new y value is a sum of products of the coordinates with the sine terms of the related axis vectors.*

3.5.2 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]]);
```

This is the realistic amount of code to write to transform all points from 3-D to 2-D.

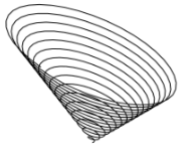

Figure 13: A conical helix (t/2*Math.cos(t), t*Math.sin(t), t) shown as (x,y,z)=f(t) with implement.html on a Canvas2DRenderingContext testing the javascript example code.

4 Important proofs of the transformation behaviour

A very important thing is to show, that the linearity of the transformation is in order. With a wrong function, bad thing can happen. With the right functions, linear combinations should stay in the subspace.

4.1 The origin stays in the origin

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

Proof:

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

4.2 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.

Proof:

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

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

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

4.3 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:

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

4.4 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:

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

4.5 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.

5 Corollaries

5.1 Converting four Dimensions down to two dimensions

The proposed 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 by giving each dimension a direction vector around the unit circle and relying the horizontal and vertical amounts of cosine and sine. For Example, 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 so be used in a general $m \times n$ case, where one goes from n dimensions down to m dimensions.[3]

Remark. The other directions to add a new dimension is more difficult, if you have to split the numbers up and have less equations than variables. For the best reconstruction of the three dimensions i am currently between guessing how, researching and preparing SVD, and will provide information, when they are ready.

5.2 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 $\vec{w} \in W$, $W \subset R^2$.

This operation can also be extended into any finite number of dimensions, and will result in two coordinates then. Just add the dimensions to $\vec{v}, \vec{c}, \vec{s}$ and see.

Proof:

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

Meanwhile it is clear, that this operation is the same as $\nabla\vec{f}(\vec{x}) \cdot \vec{v}$, which is the natural dot product of the gradient vector of our linear functional $\vec{f}(\vec{x})$ with the coordinate vector.

6 Derivatives of $\vec{f}(\vec{x}) : V \to W$

6.1 Derivative

Again we begin with $\vec{f}(\vec{x}) : V \subset \mathbb{R}^3 \to W \subset \mathbb{R}^2$.

$$\vec{f}(\vec{x}) := \begin{pmatrix} \vec{x}_1 r_x \cos\varphi_x + \vec{x}_2 r_y \cos\varphi_y + \vec{x}_3 r_z \cos\varphi_z \\ \vec{x}_1 r_x \sin\varphi_x + \vec{x}_2 r_y \sin\varphi_y + \vec{x}_3 r_z \sin\varphi_z \end{pmatrix}$$

The first derivatives after the vector are the following by using the product rule and partial differentiation. The scalar component of the input is gone, because the derivative is 1 and the other summand of the derived product is zero, because the cosine or sine function are treated like either like a constant or like a function and become zero because there is the wrong variable to differentiate in the angle.

$$\partial_1(\vec{f}_1(\vec{x})) = r_x \cos\varphi_x$$
$$\partial_2(\vec{f}_2(\vec{x})) = r_x \sin\varphi_x$$

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

The derivatives of the angles are not taken. I thought about setting a six argument function up for, and about six component vectors. But not now.

In our derivative the slope is the axis vector. Because the point is moving by that vector. From its current position along a straight line, when multiplied. It is a linear function and the point is moving only along a line, the slope is right.

Remark. All points passed to the derived function would land on the point of the vector, because the coordinate is gone after differentiating it once. The function is kind of useless from here on, if we do not utilize it another way. There are possibilities, i will show some already.

$$\begin{aligned}\partial_1(\vec{f}(\vec{x})) &= \vec{e}_x \\ \partial_2(\vec{f}(\vec{x})) &= \vec{e}_y \\ \partial_3(\vec{f}(\vec{x})) &= \vec{e}_z\end{aligned}$$

The second derivatives are already zero, because the returned vectors are constants with respect to the taken input variable.

In a different meaning, there is no second derivative, because the coordinate system is linear. It is a straight line. It has no curves, so no tangent. There is no curvature, so no second derivative. But it is perfect. And calculus is right, because the three vectors are three straight lines. When multiplying the axes with the coordinates, the point moves along straight lines.

$$\begin{aligned}\partial_1^2(\vec{f}(\vec{x})) &= 0 \\ \partial_2^2(\vec{f}(\vec{x})) &= 0 \\ \partial_3^2(\vec{f}(\vec{x})) &= 0\end{aligned}$$

Conclusion. The first derivatives represent the axis vectors. The gradient gives us the complete coordinate system back, but in a different order.

If we use the gradient then in another composition (with matrix vector multiplication with a three coordinate vector), we can apply the mapping again.

$$\nabla\vec{f} := \begin{pmatrix} \vec{e}_x \\ \vec{e}_y \\ \vec{e}_z \end{pmatrix}$$

If we transpose the column vector again, we get a row vector, which contains the three vectors of the coordinate system.
d

$$(\nabla\vec{f})^T := \begin{pmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \end{pmatrix} = \boldsymbol{A}$$

Now we could reuse the vector of vectors (the matrix) and multiply again with coordinates. But before, we come to another conclusion, which i had underways, after writing down the transposed gradient at the next morning, reading a lecture script about Analysis 2 (vector calculus).

$$(\nabla\vec{f})^T \Leftrightarrow \boldsymbol{A} \Leftrightarrow \boldsymbol{J}(\vec{f}(\vec{x})) := \begin{pmatrix} \partial_1 f_1 & \partial_2 f_1 & \partial_3 f_1 \\ \partial_1 f_2 & \partial_2 f_2 & \partial_3 f_2 \end{pmatrix}$$

The transposed gradient of the vector function $\vec{f}(\vec{x}) : V \to W$ is the Jacobi Matrix, which is equal to the matrix, i discussed already. This possibly makes another re-ordering neccessary. But first look yourself.

I have set up a corollary earlier (5.2), which uses the two row vectors with the three cosines and the three sines for a dot product with the coordinate each. In the order of the gradient $\nabla\vec{f}(\vec{x})$, the axis

vectors are column vectors themselves. The vector $\begin{pmatrix} x' \\ y' \end{pmatrix}$ is a natural result of a dot product with the them.

$$\begin{aligned} \nabla \vec{f}(\vec{x}) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \begin{pmatrix} \sum_{i=1}^{3} (\nabla \vec{f}_1)_i \vec{v}_i \\ \sum_{i=1}^{3} (\nabla \vec{f}_2)_i \vec{v}_i \end{pmatrix} \\ &= \begin{pmatrix} x' \\ y' \end{pmatrix} \end{aligned}$$

Which is equal to 5.2 and of course the formula $\vec{w} = x\vec{e}_x + y\vec{e}_y + z\vec{e}_z$ again. You see some natural connections between the basic functional, our formula, the derivatives, other formulas and our other methods which result in the same planar projection.

6.2 Integral

If i sum the three integrals $\vec{f}(\vec{x}) = \int \vec{e}_x \partial x + \int \vec{e}_y \partial y + \int \vec{e}_z \partial z$. I get the function back. But we will see after integration of positive and negative values, it has to be fixed once for those case. But we will do it below.

$$\begin{aligned} \int \vec{e}_x \partial x = x \begin{pmatrix} r_x \cos \varphi_x \\ r_x \sin \varphi_x \end{pmatrix} + \vec{C}_1 = x\vec{e}_x + \vec{C}_1 \\ \int \vec{e}_y \partial y = y \begin{pmatrix} r_y \cos \varphi_y \\ r_y \sin \varphi_y \end{pmatrix} + \vec{C}_2 = y\vec{e}_y + \vec{C}_2 \\ \int \vec{e}_z \partial z = z \begin{pmatrix} r_z \cos \varphi_z \\ r_z \sin \varphi_z \end{pmatrix} + \vec{C}_3 = z\vec{e}_z + \vec{C}_3 \\ \vec{C}_1 + \vec{C}_2 + \vec{C}_3 = \vec{C} \\ \vec{f}(\vec{x}) = x\vec{e}_x + y\vec{e}_y + z\vec{e}_z + \vec{C} \end{aligned}$$

What about the vector of the integration constants, $\vec{C}$? We can solve easily. We know about the transformation, that the zero vector maps to the zero vector. But anyways we have to set the function to zero and solve for the constant vector.

$$\begin{aligned} \vec{f}(\vec{x}) &= x\vec{e}_x + y\vec{e}_y + z\vec{e}_z + \vec{C} \\ \vec{f}(\vec{0}) &= 0\vec{e}_x + 0\vec{e}_y + 0\vec{e}_z + \vec{C} = \vec{0} \\ \vec{C} &= \vec{0} \end{aligned}$$

Remark. The constant looks like a valuable extension of the function. Looks like a fixed additional translation away from the null origin to me. $x = a + Ax$ is a formula of affine transformations, with translation included, and this seems to match, too.

Corollary. $\vec{f}(\vec{0})$ equals the summed integration constant $\vec{f}(\vec{0}) = \vec{C}$. 1 *Corollary. If the result of $\vec{f}(\vec{0}) = \vec{w}$ does not equal zero ($\vec{0}$), one can solve for the constant $\vec{C}$, which is the translation from the origin. If the normal result of $\vec{f}(\vec{0})$ is $\vec{f}(\vec{0}) = \vec{0}$ then $\vec{f}(\vec{0}) = \vec{C}$.*

$$\begin{aligned} \vec{f}(\vec{x}) &= x\vec{e}_x + y\vec{e}_y + z\vec{e}_z + \vec{C} \\ &= \vec{w} = \begin{pmatrix} a \\ b \end{pmatrix} \\ \vec{f}(\vec{0}) &= 0\vec{e}_x + 0\vec{e}_y + 0\vec{e}_z + \vec{C} = \begin{pmatrix} 12 \\ 7 \end{pmatrix} \\ \vec{C} &= \begin{pmatrix} 12 \\ 7 \end{pmatrix} \end{aligned}$$

The vector $\begin{pmatrix}12\\7\end{pmatrix}$ is an example for an additional translation. By examining $\vec{f}(\vec{0})$, we can see, whether our image is shifted or not away from the origin. Because in a non-manipulated system, the origin maps to the origin.

About the constant of integration

The three integration constants can be different from zero and then add up to zero together. So there can a little more detail be hidden in $\vec{f}(\vec{0}_{\mathbb{R}^3}) = \vec{0}_{\mathbb{R}^2}$.

$$\vec{C} = \vec{C}_1 + \vec{C}_2 + \vec{C}_3$$
$$\vec{0} = \begin{pmatrix}-1\\0\end{pmatrix} + \begin{pmatrix}1\\-1\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix}$$

The good news. Currently i can not imagine a case, where we have this situation.

Integration with positive coordinates

If i take the coordinates as limits of integration from 0 to x, 0 to y, 0 to z, i probably get the transformation again. Let us try to integrate $\begin{pmatrix}3\\1\\7\end{pmatrix}$. I have had the idea in the bus, to check this possibility out, too.

$$\begin{aligned}\int_0^3 \vec{e}_x \partial x + \int_0^1 \vec{e}_y \partial y + \int_0^7 \vec{e}_z \partial z = \\ = (3\vec{e}_x + \vec{C}_1 - 0\vec{e}_x - \vec{C}_1) + (1\vec{e}_y + \vec{C}_2 - 0\vec{e}_y - \vec{C}_2) + (7\vec{e}_z + \vec{C}_3 - 0\vec{e}_z - \vec{C}_3) \\ = 3\vec{e}_x + 1\vec{e}_y + 7\vec{e}_z\end{aligned}$$

Which is then summed up as a two dimensional vector.

Integration again with negative coordinates.

Let us try to integrate $\begin{pmatrix}-3\\-1\\-7\end{pmatrix}$. After examining whether [-3,0] or -[0,3] is the right way to integrate negative coordinates, i come to the conclusion, integrate from [0,3] and subtract the integrals. If you think, i am wrong, hold on a moment.

$$\begin{aligned}-\int_0^3 \vec{e}_x \partial x - \int_0^1 \vec{e}_y \partial y - \int_0^7 \vec{e}_z \partial z = \\ = -(3\vec{e}_x + \vec{C}_1 - 0\vec{e}_x - \vec{C}_1) - (1\vec{e}_y + \vec{C}_2 - 0\vec{e}_y - \vec{C}_2) - (7\vec{e}_z + \vec{C}_3 - 0\vec{e}_z - \vec{C}_3) \\ = -3\vec{e}_x - 1\vec{e}_y - 7\vec{e}_z\end{aligned}$$

The integral needs to be fixed:

We can fix the integral, which can not be changed manually each time, by looking whether the coordinates are positive or negative. Using the absolute value in the limits of integration and the sign function in front of the integrals make the thing work automatically for any coordinate, and we could integrate over each coordinate.

$$\begin{aligned}sign(x)\int_0^{|x|} \vec{e}_x \partial x + sign(y)\int_0^{|y|} \vec{e}_y \partial y + sign(z)\int_0^{|z|} \vec{e}_z \partial z \\ = \pm|x|\vec{e}_x \pm |y|\vec{e}_y \pm |z|\vec{e}_z\end{aligned}$$

7 Projecting just z onto a vector

What i did not get before was the projection onto a vector. I wondered about how to add the third axis. We already have seen the three independent axes. Now i have found out, how to project just the z coordinate into the $\mathbb{R}^2$ system and to keep the xy-plane the same.

$$\begin{aligned}\begin{pmatrix}x\\y\end{pmatrix} + z\begin{pmatrix}r_z\cos\varphi_z\\r_z\sin\varphi_z\end{pmatrix} &= \begin{pmatrix}x+zr_z\cos\varphi_z\\y+zr_z\sin\varphi_z\end{pmatrix}\\ &= \begin{pmatrix}x'\\y'\end{pmatrix}\\ &= \vec{w}\end{aligned}$$

I can explain what is happening. By the formulas we already have seen, the two vectors are summed together. The first one carries the x and the y coordinates. The second one, multiplies the z-axis vector with the z-coordinate. Like you know from before, this moves the point along or parallel to along the z-axis vector. And of course stops at the right place.

Example JavaScript code

```
var zAngle = rad(45);
var zAxisCos = Math.cos(zAngle);
var zAxisSin = Math.sin(zAngle);
function transform(points3) {
    var points2 = [];
    var p,x,y,z;
    for (var i = 0, j = points3.length; i < j; i++) {
        p = points3[i];
        x = p[0], y = p[1], z = p[2];
        points2.push([
            x+z*zAxisCos,
            y+z*zAxisSin
        ]);
    }
    return points2;
}
```

8 Estimation

8.1 First guess

I was thinking about, how $\|\boldsymbol{A}\vec{x}\|$ and $\|\vec{x}\|$ behave.

Definitly wrong is, that $\|\boldsymbol{A}\vec{x}\| \leq \|\vec{x}\|$. The two-d position vectors get a little longer, after summing three components up each component.

But for sure, by a theorem of uniformed boundedness, and by what ive noticed already, there is the possibility, that a constant $c \in \mathbb{R}$ exists, such that $\|\boldsymbol{A}\vec{x}\| \leq c\|\vec{x}\|$.

$$\exists c \in \mathbb{R} : \|\boldsymbol{A}\vec{v}\| \leq c\|\vec{v}\|$$

I tried to choose the constant, for the first time, and thought it could be the maximum of the norm of Ax over the norm of x.

$$c := \max_{\vec{v}\in V}\{\frac{\|\boldsymbol{A}\vec{v}\|}{\|\vec{v}\|}\} = \frac{\max\{\|\boldsymbol{A}\vec{v}\|\}}{\|\vec{v}\|}$$

If i use this c, i get that the norm of any Av is less than or equal to the maximum of all norms of Av, which means, w is bounded by the largest w, since the norm of v cancels in this term.

$$\|\boldsymbol{A}\vec{v}\| \leq \frac{\max\{\|\boldsymbol{A}\vec{v}\|\}}{\|\vec{v}\|}\|\vec{v}\|$$
$$= \|\boldsymbol{A}\vec{v}\| \leq \max\{\|\boldsymbol{A}\vec{v}\|\}$$
$$= \|\vec{w}\| \leq c\|\vec{v}\|$$

Looks a bit ridiculous, because the norm is after simplification bounded by its largest value, because the two norms of v cancel the fraction. But it isnt really. I am sure, we are making progress here soon.

The theorem of uniformed boundedness is a part of functional analysis. I have not proven its correctness in my case, or not read right, but i think, this works here.

Remark. To be continued.

8.2 Bounds with r_n values TODO)

Remark. This section has no consensus. TODO

8.3 Equality of norms (TODO)

Norms are said to be equal if there are two constant c,C such that

$$c\|\vec{v}\| \leq \|\boldsymbol{A}\vec{v}\| \leq C\|\vec{v}\|$$

Remark. TODO.

9 Cauchy Sequences and Convergence

Remark. This section is not formulated.

Convergence means, that a sequence comes step by step closer together, until the sequence items come so close together, that the distance goes to zero. The sequence itself gets closer and closer to the point. When the n goes to infinity, the distance goes to zero.

For myself, i imagine it is like it is going the path of $1,\bar{9}$ and said to converge against 2.

$$\forall\epsilon > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : \|v_m - v_n\| < \epsilon$$

For every ϵ greater than 0 exists some index n_0 of the sequence. Which has not to be the first index because of the zero, but is the first n, where the distance of the sequence vector compared to the former sequence vector is smaller than our epsilon value.

The limit goes to some value, if the series converges.

$$\lim_{n\to\infty} v_n \to \vec{v}$$

The norm or the distance goes to zero, after going under epsilon at some point n_0.

$$(\|v_n - v_m\| = d(v_n, v_m)) \to 0$$

In three dimensions, all vector components of the sequence have to converge to some value. It depends on your sequence, whether it returns one vector with three components or is a vector build by three sequences.

Anyways, $(\vec{v}_n)_{n\in\mathbb{N}^+}$ has to follow the ordinary rules, that $\lim_{n\to\infty}(\vec{v}_n) = \vec{v}$. In shorthand, the sequence has to converge against the limit $\lim_{n\to\infty} v_n \to \vec{v}$. Or even shorter, that $(\vec{v}_n) \to \vec{v}$

For my 3-D to 2-D transformation, the following propositions are made by me.

First. If the sequence (v_n) converges to $\vec{v}$. Then $(\boldsymbol{A}\vec{v})$ converges to $\boldsymbol{A}\Gamma$.

$$(v_n)_{n\in\mathbb{N}^+} \to \vec{v}$$
$$(\boldsymbol{A}v_n)_{n\in\mathbb{N}^+} \to \boldsymbol{A}\vec{v}$$

Second. It is better to put the matrix outside of the parens, if you let a computer calculate this.

For the proof it is maybe neccessary, to put the $\boldsymbol{A}$ back into the parens. But practically i would like to know the rule and then take the smallest calculation.

$$(\boldsymbol{A}v_n) = \boldsymbol{A}\vec{v} = \boldsymbol{A}(v_n)$$

If i write the matrix in the parens of the sequence, i state, that the matrix is a part of the formula of the sequence. I think i may use this notation, as long as i explain it here.

$$\boldsymbol{A}(v_n) \to \boldsymbol{A}\vec{v}$$

Doesnt this also imply that the matrix times the limit yields the right values?

$$\boldsymbol{A}\lim_{n\to\infty}(v_n) = \boldsymbol{A}\vec{v} = \vec{w}$$

So we got to show, that there is a n_0 and that some series converges.

Let there be some sequence $(\boldsymbol{A}v_n)_{n\in\mathbb{N}^+}$. We start at $n = 1$ and when when the distance shrinks under epsilon, there is some n_0. The distance continues to shrink and will finally go to zero.

$$\|\boldsymbol{A}v_n - \boldsymbol{A}v_m\| \leq \epsilon, \forall m, n > n_0$$

TODO

Remark. This section is not finished, and at the change of dimensions or at the use of two different sets with different norms, the epsilon-delta version is required, too. It should say, that if the one goes below epsilon, the other goes below delta.

10 Summary

10.1 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 $\boldsymbol{A}$. 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.

[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.

11 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.

TODO.

A More vector mathematics to move into the right sections

A.0.1 K-Vectorspace

The following table shows the dimensions of euclidean spaces, and an excerpt of a few more spaces, not showing the many many spaces like L^p (Lebesgue integrable functions) and l^2 (quadratic summarizable sequences), Fock spaces, Hardy Spaces, or Sobolev Spaces from Differential Equations, because currently all that is beyond scope of this document.

Dimensions	Sets	Type	Description
0	$\{\emptyset\}$		The empty set. It is just empty.
1	$\mathbb{R}$		The one-dimensional space is a line The coordinate is a scalar.
2	$\mathbb{R} \times \mathbb{R}$	Euclidean	The best known $\mathbb{R}^2$ is the xy-coordinate system. The coordinates are written (x,y)
3	$\mathbb{R} \times \mathbb{R} \times \mathbb{R}$	Euclidean	The three dimensional $\mathbb{R}^3$ has width, height and depth like reality. The coordinates are written (x,y,z).
4	$\mathbb{R}^3 \times \mathbb{R}$	Minkowski	The four dimensional $\mathbb{R}^4$ has width, height, depth and also a time component and is related to physics.
∞	R×... × $\mathbb{R}$	Hilbert-Spaces	The infinite dimensional space belongs into the category of Hilbert-Spaces, which are normed and equipped with a dot product.

A $\mathbb{K}$-Vectorspace V over a body $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$ is a set V with the two operations $+$ and $\cdot$.

In this document, unless edited and stated, we speak about a vector space of real numbers with $\mathbb{K} = \mathbb{R}$.

Written as ring it is written as $(V, +, \cdot)$. The vector space possesses two operations.

Addition: $+ : V \times V \to V, (\vec{v}, \vec{w}) \mapsto \vec{v} + \vec{w}$
Scalar multiplication: $\cdot : K \times V \to V, (\lambda, \vec{v}) \mapsto \lambda\vec{v}$.

A K vector space V must fulfill the following axioms.

1. $\forall \vec{a}, \vec{b} \in V. \vec{a} + (\vec{b} + \vec{c}) = (\vec{a} + \vec{b}) + \vec{c}$ (associativity)
2. $\forall \vec{a}, \vec{a}' \in V. \vec{a} + \vec{a}' = 0 = \vec{a}' + \vec{a} \forall a \in V$ (additive inverse)
3. $\forall \vec{a} \in V. 1\vec{a} = \vec{a}$ (1 is an identity operator)
4. $\forall \vec{a}, \vec{b} \in V. \vec{a} + \vec{b} = \vec{b} + \vec{a}$ (commutativity)
5. $\exists \vec{0} \in V, \forall \vec{a} \in V. \vec{0} + \vec{a} = \vec{a}$ (zero element)
6. $\forall \lambda, \mu \in K, \forall \vec{a} \in V. \lambda(\mu\vec{a}) = \lambda\mu\vec{a} = \mu(\lambda\vec{a})$ (scalar associativity)
7. $\forall \lambda, \mu \in K, \forall \vec{a} \in V. (\lambda + \mu)\vec{a} = \lambda\vec{a} + \mu\vec{a}$
8. $\forall \lambda \in K, \forall \vec{a}, \vec{b} \in V. \lambda(\vec{a} + \vec{b}) = \lambda\vec{a} + \lambda\vec{b}$ (distributive)

A.0.2 Norms: Absolute values of vectors and matrices

The norm is the word for the vector length. Or better, it is the multidimensional *absolute value* of a vector. Remember from single variable calculus that $|-x| = x$ and $|x| = x$. The norm kind of does this with all values and puts them together. Our first norm used here is the euclidean norm, also known as the 2-norm, written $\|\cdot\|_2$.

The norm is returning the square root of the sum of the squares of the absolute values of the components of the measurable expression inside between the bars $\|(expr)\| = \sqrt{\sum_{i=1}^{n} |(expr)_i|^2}$ for any number of components, like two or three.

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.

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

Ok here we go again.

$$(\vec{v} + \vec{w}, \vec{v} + \vec{w})^{\frac{1}{2}} \leq (\vec{v}, \vec{v})^{\frac{1}{2}} + (\vec{w}, \vec{w})^{\frac{1}{2}}$$

This time i tried it algebraically, to first remove the root by squaring both sides.

$$(\vec{v} + \vec{w}, \vec{v} + \vec{w}) \leq (\vec{v}, \vec{v}) + 2(\vec{v}, \vec{v})^{\frac{1}{2}}(\vec{w}, \vec{w})^{\frac{1}{2}} + (\vec{w}, \vec{w})$$

Written as sum this is

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

This can be simplified. Now assume i split the left side up into three sums. And for more simplification i leave the equal (v,v) and (w,w) on both sides away, since they are summed, not multiplied. We get

$$2(\vec{v}, \vec{w}) \leq 2(\vec{v}, \vec{v})^{\frac{1}{2}}(\vec{w}, \vec{w})^{\frac{1}{2}}$$

Now i divide the two out.

$$(\vec{v}, \vec{w}) \leq (\vec{v}, \vec{v})^{\frac{1}{2}}(\vec{w}, \vec{w})^{\frac{1}{2}}$$

Now get rid of the square root by squaring it again. This is the Cauchy Schwarz inequality.

$$(\vec{v}, \vec{w})^2 \leq (\vec{v}, \vec{v})(\vec{w}, \vec{w})$$

Cauchy-Schwarz inequality There is another interesting inequality.

The Cauchy-Schwarz inequality is saying, that the absolute value of the dot product of two vectors is less or equal to the two norms of the two vectors multiplied. The sum of the component products left is smaller than or equal to the product of the two norms.

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

is often simplified to

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

Lets decode. The left side is inside bars. This is a real absolute value. Inside of the bars is the dot product of v and w. It returns a scalar, which could be positive or negative. The bars ensure, that the value is positive.

The right side is a product of two vector norms. The vector norm is the absolute value of the whole vector. Multiplied together, they result in a, you should know from school, what width times height is, a rectangle.

Squaring both sides simplifies the inequality. On the right side, the square root, which is pulled out of the measured vector‘s dot product with itself, is disappearing. This makes the calculations easier, than with a square root.

The product on the right side is larger. Or equal.

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

Remark. Think i have corrected it.

A.1 Parallelogram equation

$$2(\|v\|^2 + \|w\|^2) = \|v + w\|^2 + \|v - w\|^2$$

This equation says, that "the square sum of the parallelogram", on the left side of the equation, "equals the square sum of the four sides", on the right side of the equation. [6]

Proof:

$$\begin{aligned} 2(\sum_{i=1}^{n} \vec{v}_i^2 + \sum_{i=1}^{n} \vec{w}_i^2) &= \sum_{i=1}^{n} \vec{v}_i^2 + 2\vec{v}_i\vec{w}_i + \vec{w}_i^2 + \vec{v}_i^2 - 2\vec{v}_i\vec{w}_i + \vec{w}_i^2 \\ &= \sum_{i=1}^{n} 2\vec{v}_i^2 + 2\vec{w}_i^2 \\ &= 2\sum_{i=1}^{n} \vec{v}_i^2 + \vec{w}_i^2 \\ &= 2(\sum_{i=1}^{n} \vec{v}_i^2 + \sum_{i=1}^{n} \vec{w}_i^2) \end{aligned}$$

[6] The text in quotes is a translated citation of a german lecture script. http://page.math.tu-berlin.de/ ferus/skripten.html from Lineare Algebra I. I took the Parallelogram equation and Polarisation formla from, too.

A.2 Polarisation equation

Remark. I have written this into my linear algebra i script (**?**)[7], on the backside of the previous four pages, underways, in the subway, after solving the parallelogram equation (equation, not inequality) in less then a minute.

$$(v, w) = \frac{1}{4}(\|v + w\|^2 - \|v - w\|^2)$$

Oh, i have had written it with a real pen, one with a rubber. I am glad i have fetched the page. The formula is still interesting. It must have a simple meaning.

$$\begin{aligned}
\sum_{i=1}^{n} \vec{v}_i \vec{w}_i &= \frac{1}{4}(\sum_{i=1}^{n} (\vec{v}_i + \vec{w}_i)^2 - \sum_{i=1}^{n} (\vec{v}_i - \vec{w}_i)^2) \\
\sum_{i=1}^{n} \vec{v}_i \vec{w}_i &= \frac{1}{4}(\sum_{i=1}^{n} (\vec{v}_i^2 + 2\vec{v}_i \vec{w}_i + \vec{w}_i^2) - \sum_{i=1}^{n} (\vec{v}_i^2 - 2\vec{v}_i \vec{w}_i + \vec{w}_i^2)) \\
&= \frac{1}{4}(\sum_{i=1}^{n} 4\vec{v}_i \vec{w}_i) \\
&= \frac{1}{4}(4 \sum_{i=1}^{n} \vec{v}_i \vec{w}_i) \\
&= \sum_{i=1}^{n} \vec{v}_i \vec{w}_i
\end{aligned}$$

Why is this formula looking important to know? The dot product of v and w is the same as a quarter of v+w‘s norm squared minus v-w‘s norm squared, reading off the formula. A lecture script (**?**) says in the Satz 22 Polarisationsformel: "Das Skalarprodukt ist durch die zugehörige Norm also eindeutig bestimmt."[8] Maybe not correctly translated but meaning the same it says "So the dot product is uniquely defined by the related norm."

Remark. The Parallelogram Equation and the Polarisationformula hold both for Hilbert Spaces and must be true if the space claims to be a Hilbert space.

A.3 Matrix norms

There are a few possible ways to measure the multidimensional absolute values of a matrix.

Row wise. Sum up each row vector, and return the largest. This is the row norm.

$$\|A\|_{row} = \max_{i=1..m} \sum_{j=1}^{n} |A_{ij}|$$

For our matrix this is

$$\|A\|_{row} = \max_{i=1..m} \sum_{j=1}^{n} |A_{ij}| = \max\{\sum_{j=1}^{n} |r_n \cos\varphi_n|, \sum_{j=1}^{n} |r_n \sin\varphi_n|\}$$

Column wise. Sum up each column vector, and return the largest. This is the column norm.

$$\|A\|_{column} = \max_{j=1..n} \sum_{i=1}^{m} |A_i j|$$

$$\|A\|_{column} = \max_{i=1..n} \sum_{i=1}^{m} |A_{ij}| = \max_{j=1..3} \{|r_j \cos\varphi_j| + |r_j \sin\varphi_j|\}$$

TODO. The Frobenius norm is $\sqrt{3}$ for $r_n = 1$ summing up 6 squares of three sines and three cosines.

$$\|A\|_{Frobenius} = (\sum_{i=1..2, j=1..3} A_{ij}^2)^{\frac{1}{2}} \|A \circ \vec{v}\|_{Frobenius} = (\sum_{i=1..2, j=1..3} \vec{v}_i^2 A_{ij}^2)^{\frac{1}{2}}$$

[7] see footnote 1

[8] On page 54.

Proposition. The matrix norm of the coordinate systems matrix. 1 *The euclidean norm of our matrix, the Frobenius Norm, counting together the rows and columns, componentwise and squared, then pulling the root out of the whole sum, is, without $\vec{v}$ the square root $\sqrt{r_x^2 + r_y^2 + r_z^2}$. The sines squared and cosines squared add up to 1 each, and are a factor for the r_n^2. Each r_n^2 belongs to pair of sine and cosine. When sine and cosine squares are added, the r_n^2 has to be factored out first, it can not add up to two, what could be miscounted easily. If the coordinates x, y, z, say, the vector $\vec{w}$, are applied to the matrix and the Frobenius Norm is taken, the norm for $\|Av\|$ is $\sqrt{x^2r_x^2 + y^2r_y^2 + z^2r_z^2}$. The Frobenius norm is like the $\|\cdot\|_2$ norm taken, but counts matrix elements instead of vector elements.*

Remark. Needs definitly new formulation and not the algebraic simplification as the proposition.

$$\|\boldsymbol{A}\|_{Frobenius} = (r_x^2 + r_y^2 + r_z^2)^{\frac{1}{2}}$$
$$\|\boldsymbol{A} \circ \vec{v}\|_{Frobenius} = (x^2r_x^2 + y^2r_y^2 + z^2r_z^2)^{\frac{1}{2}}$$

Proof:
Without the elements of a vector

$$\begin{aligned}\|\boldsymbol{A}\|_{Frobenius} = (\sum_{i=1,2;j=x,y,z} \vec{e}_{ij}^2)^{\frac{1}{2}} \\ &= (\sum_{n=1}^{3} r_n^2 \cos\varphi_n^2 + \sum_{n=1}^{3} r_n^2 \sin\varphi_n^2)^{\frac{1}{2}} \\ &= (\sum_{n=1}^{3} r_n^2(\cos\varphi_n^2 + \sin\varphi_n^2))^{\frac{1}{2}} \\ &= \sqrt{r_x^2 + r_y^2 + r_z^2}\end{aligned}$$

And with applying the vector to A

$$\begin{aligned}\|\boldsymbol{A} \circ \vec{v}\|_{Frobenius} = (\sum_{i=1,2;j=x,y,z} \vec{v}_i^2\vec{e}_{ij}^2)^{\frac{1}{2}} \\ &= (\sum_{n=1}^{3} \vec{v}_n^2 r_n^2 \cos\varphi_n^2 + \sum_{n=1}^{3} \vec{v}_n^2 r_n^2 \sin\varphi_n^2)^{\frac{1}{2}} \\ &= (\sum_{n=1}^{3} \vec{v}_n^2 r_n^2(\cos\varphi_n^2 + \sin\varphi_n^2))^{\frac{1}{2}} \\ &= \sqrt{x^2r_x^2 + y^2r_y^2 + z^2r_z^2}\end{aligned}$$

Remark. Ongoing research.

A.4 Operator norms

TODO.

The operatornorm is a dependent norm. It depends on the currently used norm.

$$\|A\| = \|\phi\| := \sup\{\phi(\vec{x}) : \|x\| = 1\}$$

Remark. This is by heart. But i sure i am not through with using it correctly.

A.5 (moved b4 del) Taking the norm of $\vec{e}_n$ to obtain r_n from some existing coordinate system

Remark Maybe the use case is too unrealistic

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 $\mathbb{R}^3$ and the $\mathbb{R}^2$ 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 or getting the absolut values 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.

A.6 Dot product

The dot product, scalar product or inner product. It is the most important vector vector multiplication defined in space. It makes calculations with angles and detection of orthogonality possible.

It is the sum of the vector component products with either itself, or another vector.

If you pull the square root out of the dot product with a vector and itself, you obtain the current length of the vector. Dividing the vector by it‘s length will normalize the vector to a length of 1. $\|\vec{x}\| = 1$ is called the unit length. The formula and proof of the normalization of a vector is in E.1

$(\vec{v}, \vec{w})$ 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.

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

Algebraic simplifications and linear combinations.

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

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

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

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

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 obtain by scratching current column and the first row. You multiply the determinant with i, j, or k.Which

$$\vec{a} \times \vec{b} = (a_2b_3 - a_3b_2)\vec{i} - (a_1b_3 - a_3b_1)\vec{j} + (a_1{}_b2 - a2b_1)\vec{k} = \vec{c}$$

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

First i could not make out, what to proof now. But i can orient myself with (1). The proof works like this: You have to prove, that $(v \times w) \cdot v = 0$, so that $(v \times w) \perp w$ and that $(v \times w) \cdot w = 0$ and also $(v \times w) \perp w$. I will calculate this alone, without looking again. Just calculate out the cross product and multiply with the components of the one vector you dot. After rearranging the terms, the result must be zero.

$$\left(\begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \vec{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} \vec{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \vec{k} \right) \cdot \left(\vec{a}_1\vec{i} + \vec{a}_2\vec{j} + \vec{a}_3\vec{k} \right) =^{?} 0$$

and

$$\left(\begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \vec{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} \vec{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \vec{k} \right) \cdot \left(\vec{b}_1\vec{i} + \vec{b}_2\vec{j} + \vec{b}_3\vec{k} \right) =^{?} 0$$

A.8 Normal vectors

Are perpendicular to the surface, a curve or another vector and give the orientation.

For 2-D space, the normal vector of the standard basis $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is obtained by multiplying the vector with the standard normal matrix $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$.

$$\boldsymbol{N}_{\mathbb{R}^2} := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

You should multiply the 2x2 matrix with some vector and you will get a perpendicular vector back.

$$\vec{n} = \boldsymbol{N}\vec{w} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -b \\ a \end{pmatrix}$$

Proof. The resulting normal vector has to be perpendicular to the source vector, so their dot product must be zero.

$$\vec{n} \cdot \vec{w} = \sum_{i=1}^{2} \vec{n}_i\vec{w}_i = -ab + ab = 0$$

For three dimensional vectors the proof is similar, and the dot product should also return zero for a proof. A perpendicular vector is obtained by permuting the identity matrix (standard basis) and changing sign.

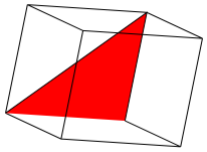

Figure 14: TODO. Can we find a simple triangulation or just simplification to use the fill() method on strictly convex sets?

A.9 Convex sets

Convex sets contain the path between two points inside the set. That means, that the direct way from a to be is not crossing the borders of the set. This can be imagined with real borders of a set. A round set, or a rectangle have all points inside together with their path. A star for example, where you have two points at two of the tips is not convex, the direct path, the straight line would cross the border of the star, leave the star, and reenter the star. There is a formula, which gives the points on the path.

$$u = \lambda\vec{x} + (1-\lambda)\vec{y}, \lambda \in [0,1], x, y \in V \implies u \in V$$

The factor lambda lies between 0 and 1. If $\lambda = 0.1$ then is $(1-\lambda) = 0.9$, if $\lambda = 0.2$ then is $(1-\lambda) = 0.8$ and so on. The result is lying on the straight line between $\vec{x}$ and $\vec{y}$. I have taken two vectors $\vec{x}, \vec{y} \in R^2$ and drawn them. The points lie on the line between the two points.

There is an inequality, which convex functions have to satisfy. And you may know it. It is the *triangle inequality*

$$\|\lambda\vec{x} + (1-\lambda)\vec{y}\| \leq \lambda\|\vec{x}\| + (1-)\|\vec{y}\|$$

or what expresses, that a function is convex.

$$f(\lambda\vec{x} + (1-\lambda)\vec{y}) \leq \lambda f(\vec{x}) + (1-)f(\vec{y})$$

What is again the *triangle inequality.*

Our image is very precise, because we designed a real coordinate system. What is happening to our new vector?

$$\lambda\boldsymbol{A}\vec{x} + (1-\lambda)\boldsymbol{A}\vec{y} = \boldsymbol{A}(\lambda\vec{x} + (1-\lambda)\vec{y})$$

By the rule of linearity.

Remark. To be studied within the next days.

B Definition of $\mathbf{R}^{2\times3}$!

The $\mathbb{R}^{2\times3}$ is NOT a K-vectorspace. The axioms of a body and a vector space can not be fulfilled.

But the $\mathbb{R}^{2\times3}$ is a vector space. A vector space defines addition and scalar multiplication.

And we have both. $Av + Az = A(v+z) = w$, and $(v) = A()$ in the previous definitions.

Can you spot one difficulty in our vector space?

$$\lambda\vec{v} = \lambda\vec{w}$$

But it is a vector space? Yes. But note, that also

$$\vec{u} + \vec{v} = \vec{w} + \vec{x}$$

With that i can certainly announce that, under human conditions and rational thoughts, that

Proposition. The $\mathbb{R}^{2x3}$ is the vector space of 2-D images made of 3-D coordinates. 1 *For the human nature the most natural use of the $\mathbb{R}^{2x3}$ is to produce 2-D images of 3-D coordinates with. It is possible to configure the 2x3 space for other calculations, but the most reasonable, because of the perfect images is to translate 3-D points into 2-D points with. The $\mathbb{R}^{2x3}$ is the vector space of 2-D images made of 3-D coordinates.*

There is a point about topology i did not clear yet, except for myself. The 2-D image covers points from far back in the 3-D. They lie on a straight line which is normal to the plane. And there are infinite lines. But not anywhere must be covered points.

Proposition. The $\mathbb{R}^{2x3}$ is the vector space of 2-D images made of 3-D coordinates. 2 *In the topology of the 2-D image, certain points on the plane represent more than one point in the image. All covered points lie on a straight line, which is normal to the plane.*

Because of that problem there is no bijectivity, no inverse. Here is a problem, only approximations and knowledge about the object projected can approximate or approximatly solve.

Proposition. The $\mathbb{R}^{2x3}$ is the vector space of 2-D images made of 3-D coordinates. 3 *There is no bijection. Because there is no injection because of the covered points.*

Now i come to the part, to say so, that the 2x3 basis above is the correct one for the R2x3, by saying before, that the most natural is, to produce 2-D images of three coordinates with.

Proposition. The $\mathbb{R}^{2x3}$ is the vector space of 2-D images made of 3-D coordinates. 4 *The most natural configuration for a 3-D coordinate system in the 2-D space is the arrangement of the axes around the unit circle. The cosine and sine orthonormal standard basis vector (r cos phi, r sin phi) is the vector to choose for each of the three axes. The unit length of the axes can be chosen by modifying the value of r. This is exactly the behaviour of polar coordinates. The vectors of the axes are the values of the coefficients for the algebra transforming the three coordinates into the two.*

Which makes another thing clear. The linear combination of the axes is optimal.

Proposition. The $\mathbb{R}^{2x3}$ is the vector space of 2-D images made of 3-D coordinates. 5 *The three axes of a two dimensional image of a three dimensional coordinate systems are linear dependent in terms of the resulting 2-D image, concerning the result of the $\mathbb{R}^{2\times3}$ algebra being interpreted as a subset of the regular $\mathbb{R}^2$ vector space. But for the conversion of the coordinates the linear combination is optimal.*

The basis of $\mathbb{R}^{2\times3} is arranged around a circle. The basis of \mathrm{R}^2$ and $\mathbb{R}^3$ are linearly independent. The basis of $\mathbb{R}^{3\times4} for example would be arranged around a sphere.$

Remark. To be continued.

B.1 Orthogonality in the 2x3 matrix

The thing consists of three horizontal moves to produce the new x. And of three vertical moves to produce the new y. Or in other words. It does not know about horizontal or vertical. It combines the three input values into one new by applying algebraicly coefficients to the input, or other way round by applying the input to the basis of the vector space, and returns a new 2-D vector, which can be

interpreted by $\mathbb{R}^2$ or will be interpreted as flat $\mathbb{R}^{2\times3}$ image with three axes around a circle.

Remark. To explain, that the R2 standard basis does the orthogonal interpretation of the 2-D x and y coordinates.

Remark. Another good question to myself. Has R2x3 a standard basis of R2 for the result and a "2x3 basis" for the linear map? Does it come as double pack?

C Deconstructing the 2x3

The whole section about R2x3 is to be continued very soon

D deprecated 2x3 todo

D.1 2x3 standard basis

Remark. This example concerning the book (3) seems to be about R6 and not match my interpretation.

A 2x3 basis is a set of six matrices being constructed out of five basis vectors. In the Appendix of (3), there is a basis of a 2x3 matrix printed, together with a few arguments. The professor constructed a 2x3 standard basis by multiplying the 2-D standard basis vectors with the 3-D standard vectors.

$$\mathrm{e}_1^{\mathbb{R}^2} = \begin{pmatrix}1\\0\end{pmatrix} e_2^{\mathbb{R}^2} = \begin{pmatrix}0\\1\end{pmatrix}$$
$$\mathrm{e}_1^{\mathbb{R}^3} = \begin{pmatrix}1\\0\\0\end{pmatrix} e_2^{\mathbb{R}^3} = \begin{pmatrix}0\\1\\0\end{pmatrix} e_2^{\mathbb{R}^3} = \begin{pmatrix}0\\0\\1\end{pmatrix}$$

Multiplying $e_i^{\mathbb{R}^2}$ with $(e_j^{\mathbb{R}^3})^T$ yields six independent matrices, which, when added, form the standard basis.

$$E^{\mathbb{R}^{2\times3}} = \begin{pmatrix}1&0&0\\0&0&0\end{pmatrix} + \begin{pmatrix}0&1&0\\0&0&0\end{pmatrix} + \begin{pmatrix}0&0&1\\0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0\\1&0&0\end{pmatrix} + \begin{pmatrix}0&0&0\\0&1&0\end{pmatrix} + \begin{pmatrix}0&0&0\\0&0&1\end{pmatrix}$$

My task is now, to deconstruct our formula, to meet the requirements.

$$\mathrm{e}_1^{\mathbb{R}^2} = \begin{pmatrix}u\\0\end{pmatrix} e_2^{\mathbb{R}^2} = \begin{pmatrix}0\\v\end{pmatrix}$$
$$\mathrm{e}_1^{\mathbb{R}^3} = \begin{pmatrix}a\\0\\0\end{pmatrix} e_2^{\mathbb{R}^3} = \begin{pmatrix}0\\b\\0\end{pmatrix} e_2^{\mathbb{R}^3} = \begin{pmatrix}0\\0\\c\end{pmatrix}$$
$$\mathrm{E}^{\mathbb{R}^{2\times3}} = \begin{pmatrix}ua&0&0\\0&0&0\end{pmatrix} + \begin{pmatrix}0&ub&0\\0&0&0\end{pmatrix} + \begin{pmatrix}0&0&uc\\0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0\\va&0&0\end{pmatrix} + \begin{pmatrix}0&0&0\\0&vb&0\end{pmatrix} + \begin{pmatrix}0&0&0\\0&0&vc\end{pmatrix}$$

Here we can subtitute

$$ua = r_x\cos\varphi_x, va = r_x\sin\varphi_x, ub = r_y\cos\varphi_y, vb = r_y\sin\varphi_y, uc = r_z\cos\varphi_z, vc = r_z\sin\varphi_z,$$

It should result in

$$\begin{pmatrix} r_x\cos\varphi_x & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & r_y\cos\varphi_y & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & r_z\cos\varphi_z \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ r_x\sin\varphi_x & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & r_y\sin\varphi_y & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & r_z\sin\varphi_z \end{pmatrix} = \begin{pmatrix} r_x\cos\varphi_x & r_y\cos\varphi_y & r_z\cos\varphi_z \\ r_x\sin\varphi_x & r_y\sin\varphi_y & r_z\sin\varphi_z \end{pmatrix}$$

Remark. My first conclusions, while inventing the chapter and the excercise. To be removed.

From trigonometry, we know, that cosine and sine form a right angle. They are perpendicular to each other. But taking the norm of the vector results in r, not in 0, because $sin^2 + cos^2 = 1$. $r^2\sin^2 + r^2\cos^2 = r^2$. This is not orthogonal in terms of the dot product, which should result in 0, if the angle is 90 degrees.

For simplification we should do a few things. First, we assume $r_x = r_y = r_z = 1$. Without a factor in front of sine and cosine, they both together have unit length of one. With $\| \begin{pmatrix} \frac{r_x\cos\varphi_x}{r_x} \\ \frac{r_x\sin\varphi_x}{r_x} \end{pmatrix} \| = 1$ the vector is normalized.

Secondly, we substitute the cosine and sine terms with simple letters. Sine, cosine and r form a triangle. Thinking of a triangle, and sin = opposite/hypotenuse and cos = adjacent/hypotenuse, will make it easier for us to find the factors and to solve this system.

$$cos = \frac{adjacent}{hypotenuse} sin = \frac{opposite}{hypotenuse}$$

D.2 Lefthanded and righthanded system

Without calculating them, but quick-concluding them from the existing, i get the following matrices, which look like bases, containing only zeros and ones. With the quick issue, that the length of the columns with two ones is square root of two, while a zero-one vector has length one.

$$\mathrm{K} = \mathrm{R}E^{\mathbb{R}^{2\times3}}_{lefthand} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$E^{\mathbb{R}^{2\times3}}_{righthand} = \begin{pmatrix} -1 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}$$

Remark. This section is incomplete, almost meaningless yet, and better not to be taken as is.

Remark Proof or contradictions TODO

Remark. August 5. I will work on this chapter the next days. Proving it is a 2-D image of 3 input coordinates.

E Vector space tools

E.1 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. Whenever you wish or need to reduce or enlarge a vector to unit length, that $\|\vec{v}\| = 1$ or $\|\vec{w}\| = 1$.
you can do this yourself,too.

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

Together with updating the vector or creating a new vector, you just have to divide the old vector components by the old vectors length. See the proof for details Taking the norm then, results in 1.
Proof:

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

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

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

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

E.2 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. The absolute value function $|\pm n| = n, \pm n \in \mathbb{R}$ makes sure

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

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

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

F Proving more rules of the main formula

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.

F.1 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.

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

In the 2x2 matrix $\boldsymbol{A}\boldsymbol{A}^T$ is $a_{ij} = a_{ji}$.
I will abbreviate $\cos\varphi_n$ with C_n and $\sin\varphi_n$ with S_n.

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

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

Also in the 3x3 matrix $\boldsymbol{A}^T\boldsymbol{A}$ is $a_{ij} = a_{ji}$.
A little bit refined the 3x3 matrix becomes this. Also, to forget about r_n is $r_n = 1$.

$$\boldsymbol{A}^T\boldsymbol{A} = \begin{pmatrix} 1 & \sin(\varphi_x + \varphi_y) & \sin(\varphi_x + \varphi_z) \\ \sin(\varphi_x + \varphi_y) & 1 & \sin(\varphi_y + \varphi_z) \\ \sin(\varphi_x + \varphi_z) & \sin(\varphi_y + \varphi_z) & 1 \end{pmatrix}$$

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 $P = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$.

The determinant of the A 2x2 determinant and inverse have the following formulas. A 3x3 determinant and inverse have the following formulas.
TODO

Without checking the importance of the transposes in the first three weeks since i started this document, i kept this begun calculation. Meanwhile i figured out why:
$((A^TA) - \lambda I) = 0$ and $((AA^T) - \lambda I) = 0$ want to be solved, and then we want the SVD.

F.1.1 Crossing three 2x2 matrices (experiment)

Remark. What is the outcome if we cross compute with three 2x2 matrices and take the differences. Can we estimate the original coordinates except for overlays? Like the cross product, we could split the 2x3 into three 2x2 matrices. Then we can take the differences.

F.1.2 Singular Value Decomposition

TODO

This a mxm orthogonal times nxm diagonal times nxn orthogonal. To get the orthogonal we have to take the eigenvectors from the products with the transpose. TODO.

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

Remark. A rectangular matrix has no eigenvalue equation. But there is a singular value decomposition, which can tell some proper things about the matrix. For that, the eigenvalues are taken from the products with the transpose. And then the decomposition continues.

The A^TA and AA^T are needed for the SVD. First we extract the eigenvalues and eigenvectors of the symmetric square matrices. Then we build So with the last two chapter we are fine.

Remark. Nice to get some excercise for this topic, which can not be solved from an outsider without any good excercise or motivating lecture. I think this excercise is fun enough.

F.1.3 The pseudo-inverse A^+ and least squares

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

G Computer Error Estimation

TODO

Remark. The roundoff error of the floating point and the condition of a matrix is basic knowledge for every computer class.

H 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. This works on any graphics surface you can connect 2-D points or draw then directly.

Remark. Missing. The fill algorithm (the stuff is pretty long) and our view frustum (). Plus the remark, we do not try to replace computer graphics. But for small visualizations it is a quick tool, for handwritten code.

H.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. It is just a translation.

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

This has to be applied to every point.

H.2 Translation

You simply add the translation vector to each point.

Remark. This is the same like shifting the origin, but translation has a meaning, that it is then done, maybe a few times, with animation, to move from a to b.

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

Remark. A affine combination is written $x = a + Ax$. So to say, the same for the origin.

H.3 Scaling

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

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

H.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;
```

H.5 Local 3x3 basis for change of units and per object rotation

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) standard basis for the $\mathbb{R}^3$ and the result is the same as if you do not apply any basis to the object, as the assumed default coordinate system in $\mathbb{R}^3$ 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.

H.5.1 Creating a 3x3 basis with cross products

```
function cross(a,b) {
    // does not multiply with the ijk components, diy
    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);
// w = [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);
// w = [0,0,-1]
```

H.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;
```

H.7 Frustum and Perspective

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

TODO

H.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;
}
```

H.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;
}
```

H.10 Metric

The distance function gives us the distance between 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);
}
```

H.11 Code Example

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

These functions are not optimized for speed. Each of the -all functions take the whole set of points, so the complexity rises to n times iterating over the point sets. To create faster code, you got to inline your code and do everything in the right order in one loop. Anyways, that is not difficult and intuitive to programmers.

```
(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;
    }
}

function compareAll(points3, points2, callback) {
    var results = [];
    for (var i = 0, j = points3.length; i < j; i++) {
        results.push(callback(points3[i],points2[i]));
    }
```

```
    return results;
}

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.compareAll = compareAll;
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;

    i++;

}
```

I How i was wrong the first time. Turning xy-around and adding z in.

Remark. This was the transformation i found before i invented the axes. This brought me to the invention.

Before i figured out, how to create three axes, i did it wrong. I wanted to turn three axes around, but did only rotate the xy-plane. With the well known formula for rotating around the z-axis in the $\mathbb{R}^2$. I tried to combine this for the other axes, but failed. But i had an idea, as the xy-axes were pointing downwards. I was writing it on the computer, the hardware y-axis was free after rotation. I added the z-coordinate to the y coordinate, believing it would shift the point upwards vertically. I found an interpretation of the righthanded coordinate system.

1. 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.
2. Now the x-axis and the y-axis point downwards. This means, the real y-axis on the real 2-D coordinate system is now 'free'.

3. So add the z-coordinate in by just adding it to y.

I will show it again now step by step.
The angle to turn each point around.

$$\angle\alpha = 1.25\pi$$

The rotation matrix for two dimensions, turns the plane around the z-axis which points invisible orthogonally out of the screen.

$$\boldsymbol{R} = \begin{pmatrix} \cos\angle\alpha & -\sin\angle\alpha \\ \sin\angle\alpha & \cos\angle\alpha \end{pmatrix}$$

v is the input vector, and w is the output vector.

$$v = (x, y, z)^T, w = (v_1, v_2)^T = (x, y)^T$$

Then we rotate each point containing only the x,y coordinate of the triple v.

$$\boldsymbol{R}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} = \vec{w}'$$

The x,y image is pointing downwards. I add each points z coordinate now to y, which moves the point upwards by the amount of z.

$$\begin{pmatrix} x' \\ y' + z \end{pmatrix} = \vec{w}''$$

After this i discovered adding three proportional parts for each coordinates and decided quickly to use cos and sin and the circle to create axes by angles.

Example
This is the EcmaScript 5 code example for rotating xy and addign z in.

```
function transform(points3) {

    var angle = 1.25*Math.PI;  // 225 deg
    var cos = Math.cos(angle), // prevent from
        sin = Math.sin(angle); // repeating calls
    var points2 = [];          // new points
    var p;                     // current point
    var x,y,u,w,z;             // coordinates

    for (var i = 0, j = points3.length; i < j; i++) {

        p = points3[i];
        u = p[0],
        v = p[1],
        z = p[2];

        // rotate x and y
        x = cos*u -sin*v;
        y = sin*u +cos*v;

        // add z vertically
        y += z;
        // could do it on one line with the previous y

        // add new point to list
        points2.push([x,y]);
    }

    return points2;
}
```

J Declaration of authorship

Remark. Have to fetch the english version of the Selbständigkeitserklaerung over the internet.

With this here i promise, i invented the thing alone, and did copy no proof nowhere and used only the books and the lecture scripts for reference i acknowledge in the bibliography.

Remark. Got to fetch the formal version of this declaration.

K License

Remark. Formulation lacks. Obviously a standard public domain license will work.

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

, , Lecture Script, Topology (english),

, Lecture Script, Functional Analysis (english),

TODO, Lecture Script,

TODO, Lecture Script,

Dirk Ferus, TU-Berlin, em., Lecture Script, Lineare Algebra 1+2, 2000, http://page.math.tu-berlin/ ferus/skripten.html

Franziska Kühn, Technische Universität Dresden, Lecture Script, Lineare Algebra und analytische Geometrie I+II, http://fkuehn.de/download/LAAG.pdf

Petra Wittbold, TU-berlin, Lecture Script, Funktionalanalysis I, http://www3.math.tu-berlin.de/Vorlesungen/SS09/FA1/Doc/Funkana1-SS06-08.06.09.pdf

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