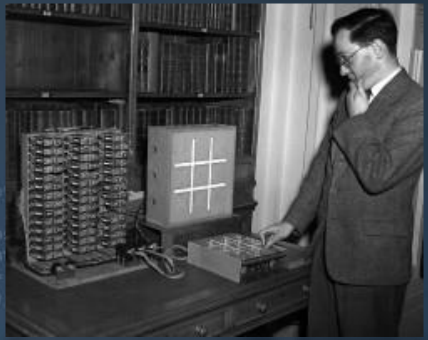
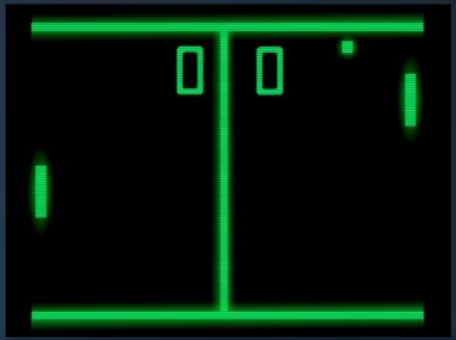
INFOGR – Computer Graphics

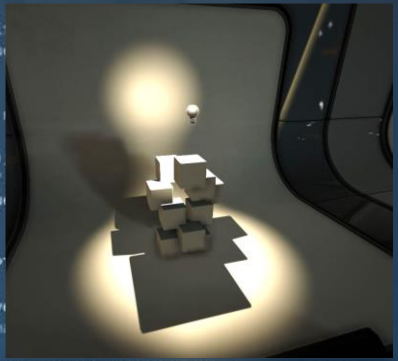
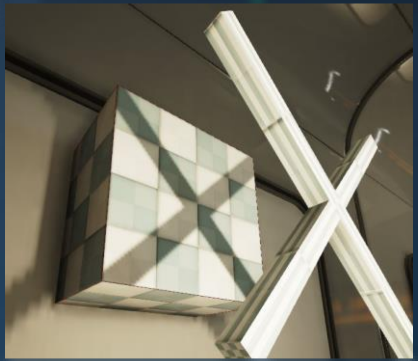
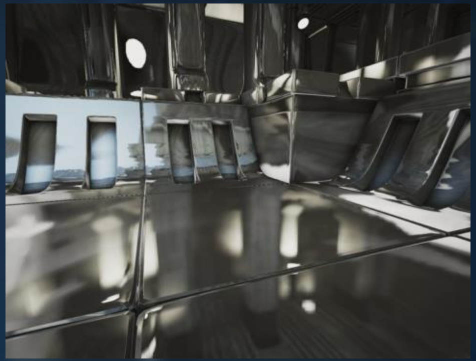
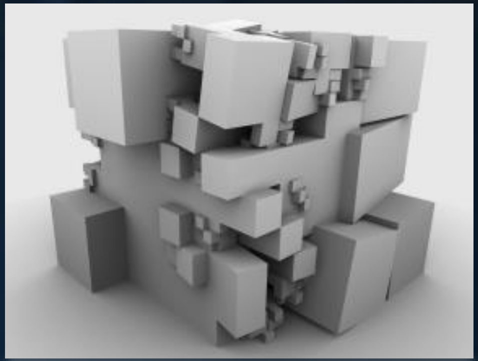
Jacco Bikker - April-July 2015 - Lecture 2: “Graphics Fundamentals”

Welcome!

Synchronize



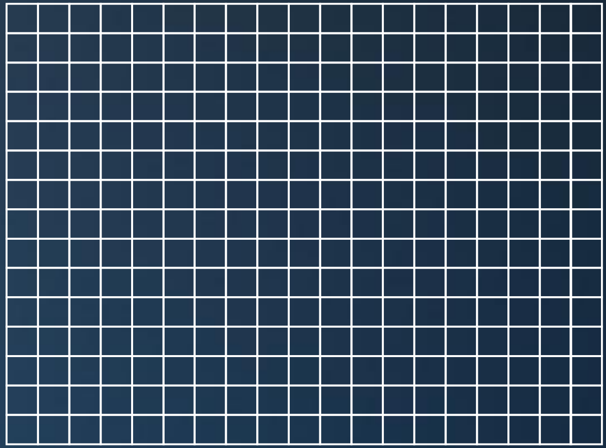

http://www.cs.uu.nl/docs/vakken/gr

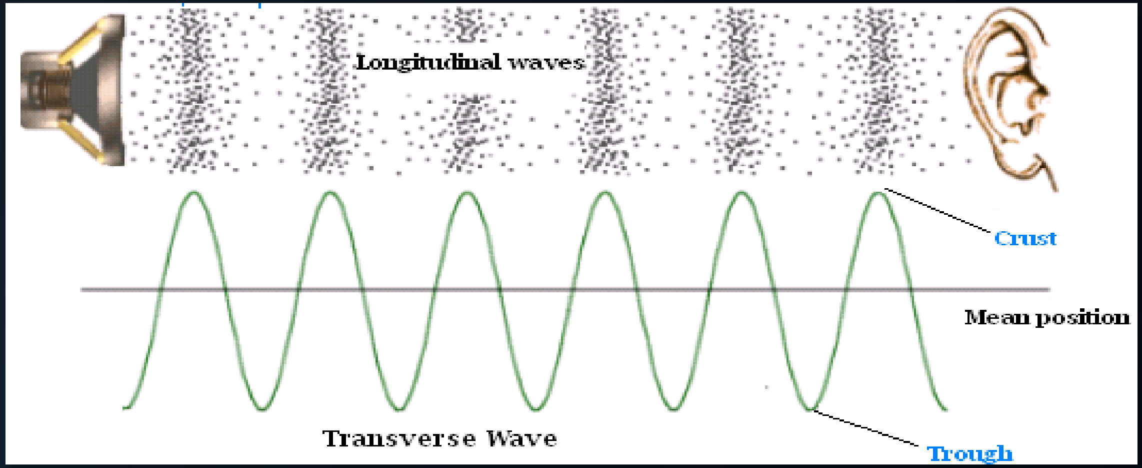
Today's Agenda:

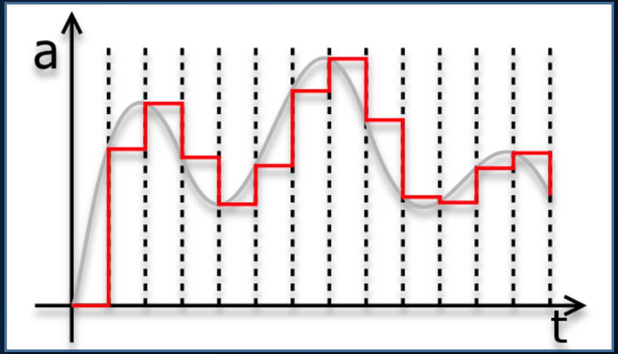
- The Raster Display
- Vector Math
- Colors

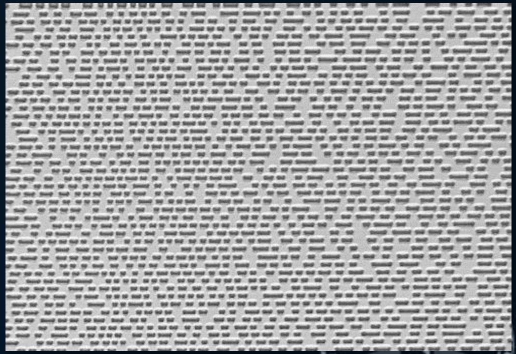
Raster Displays

Discretization



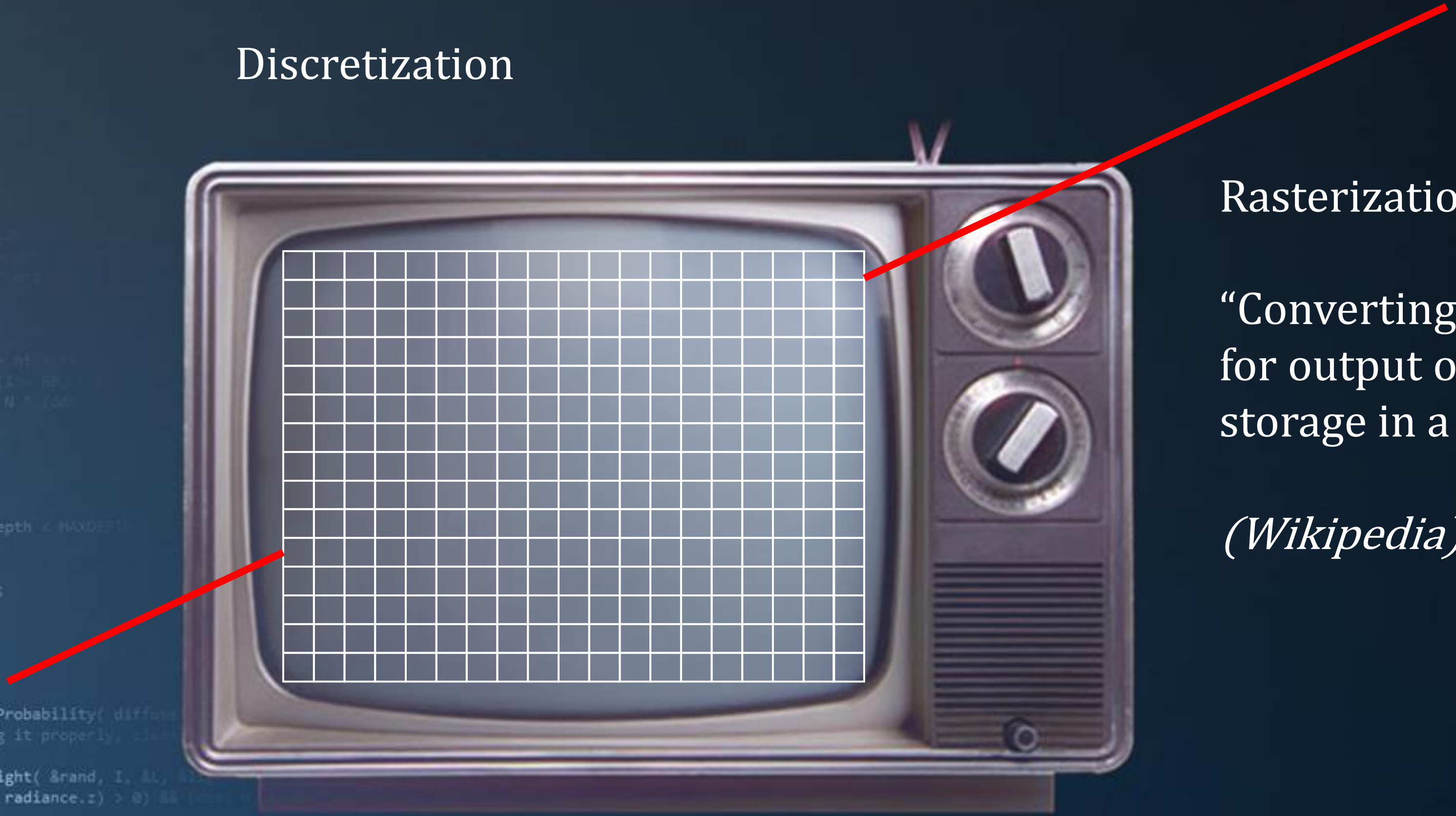






Raster Displays

Discretization



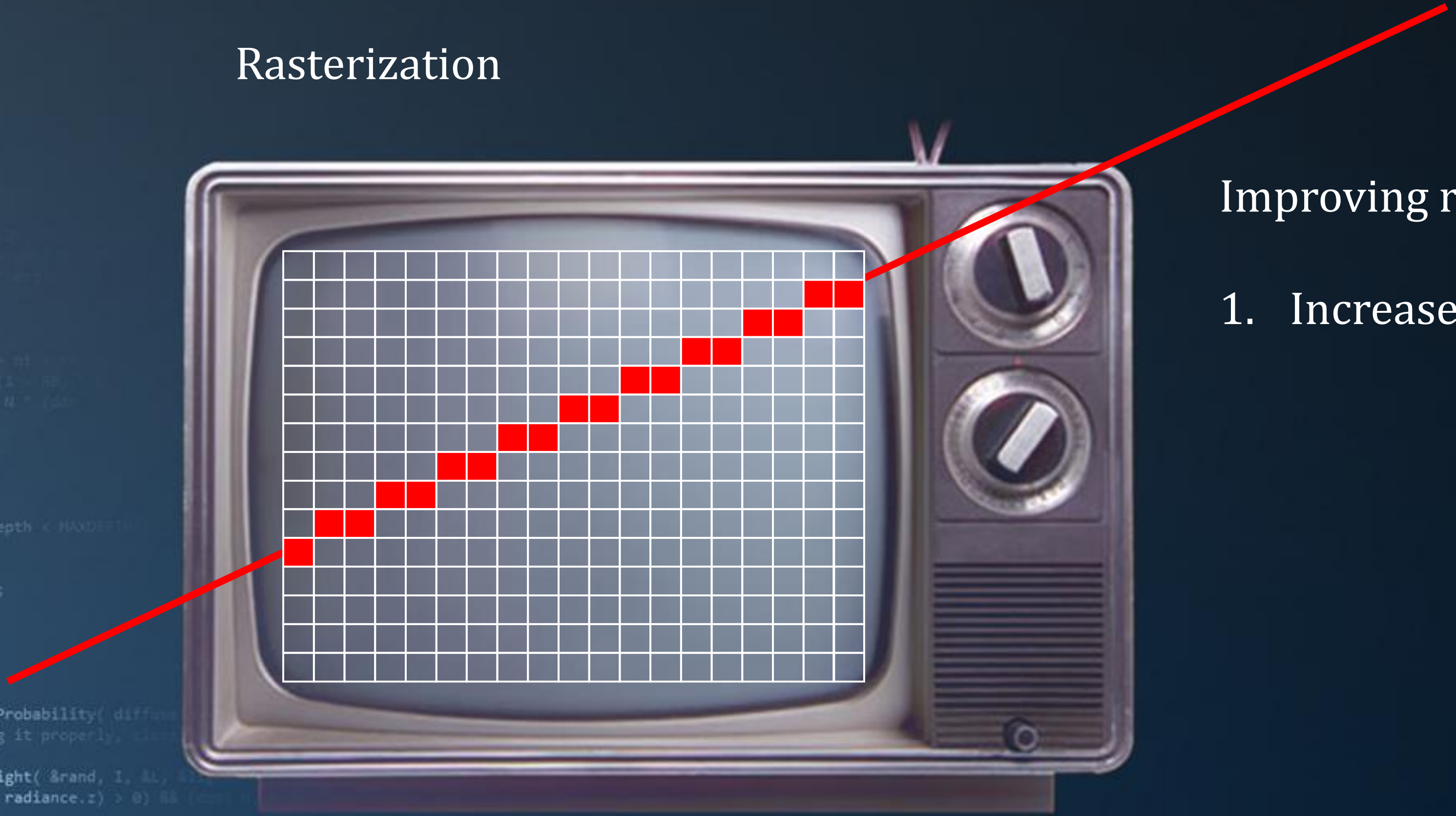
Rasterization:

“Converting a vector image into a raster image for output on a video display or printer or storage in a bitmap file format.”

(Wikipedia)

Raster Displays

Rasterization

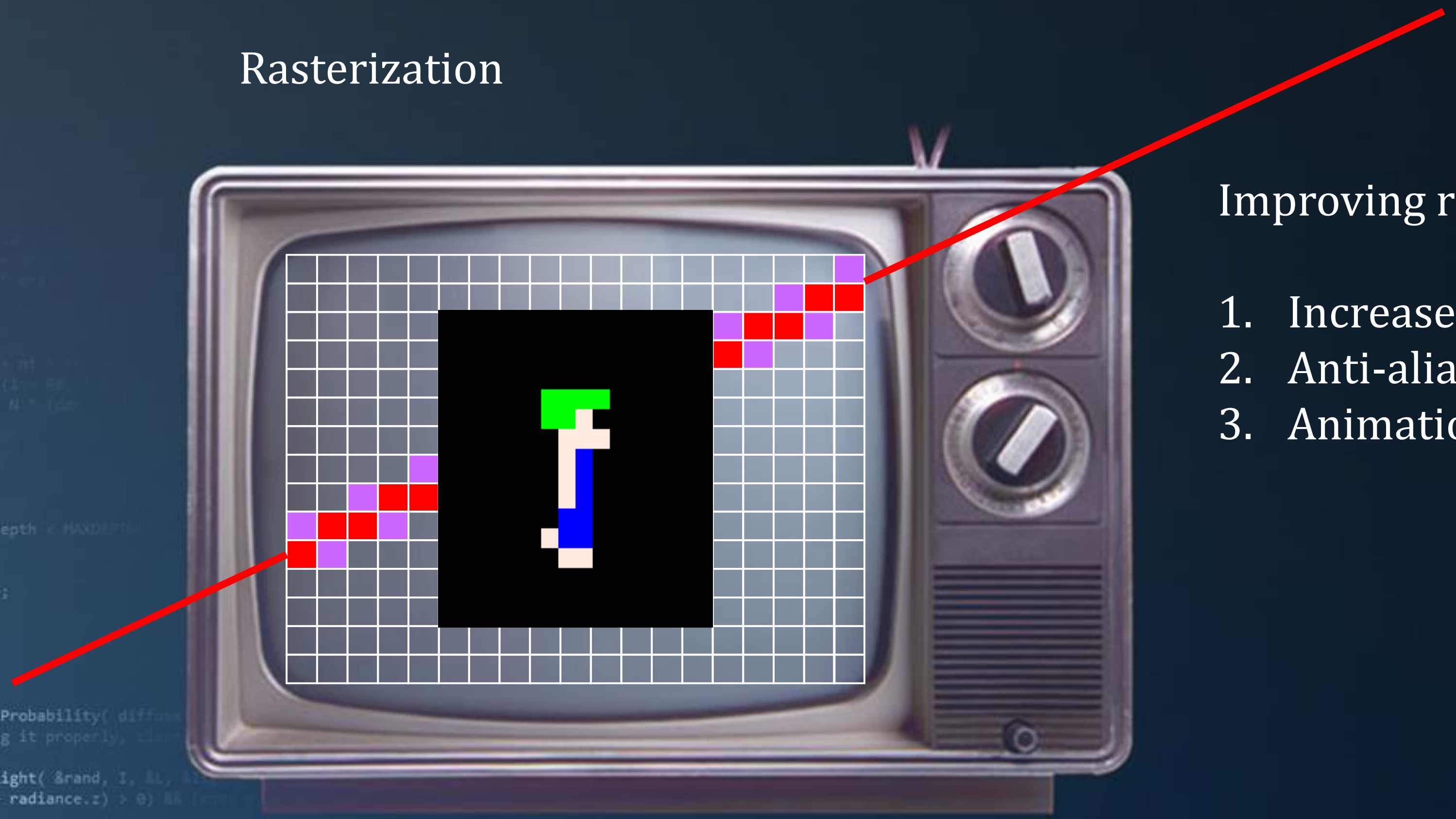


Improving rasterization:

1. Increase resolution;

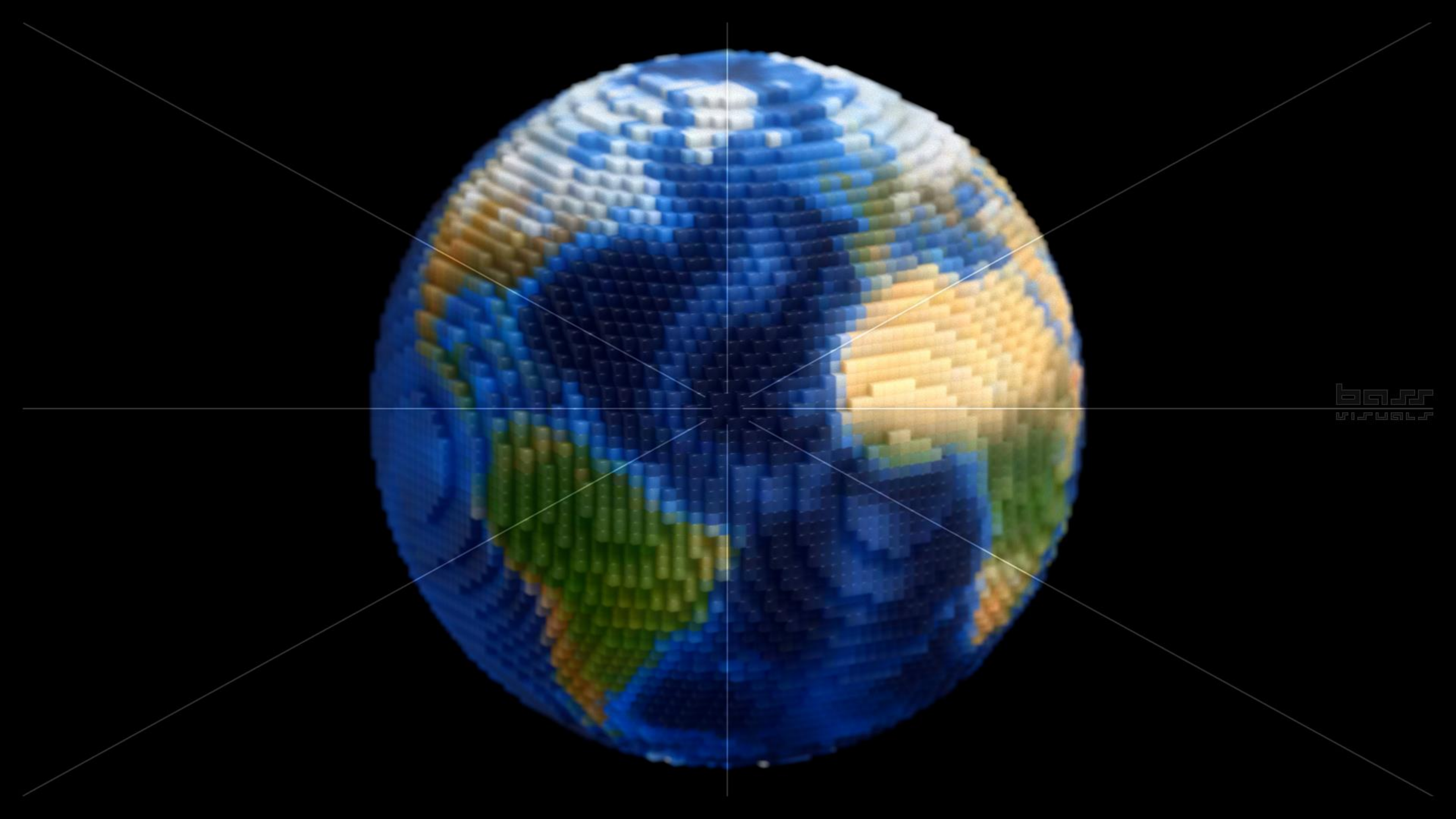
Raster Displays

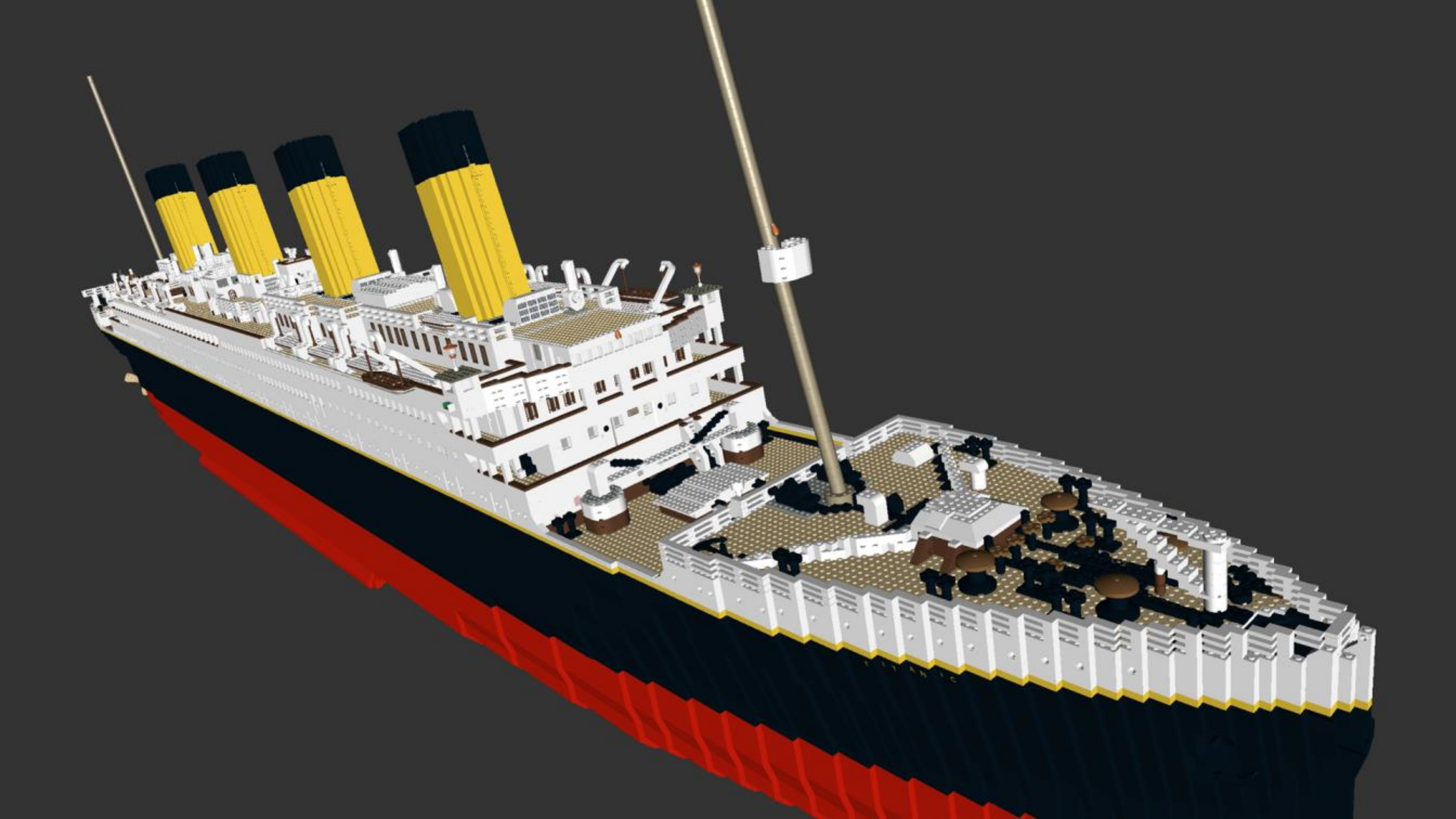
Rasterization



Improving rasterization:

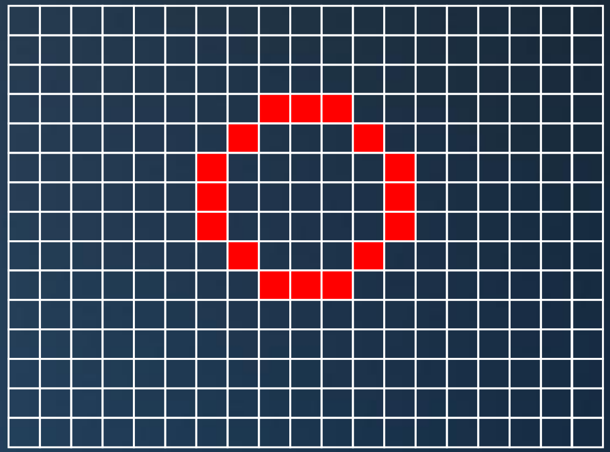
1. Increase resolution;
2. Anti-aliasing;
3. Animation.





Raster Displays

Discretization

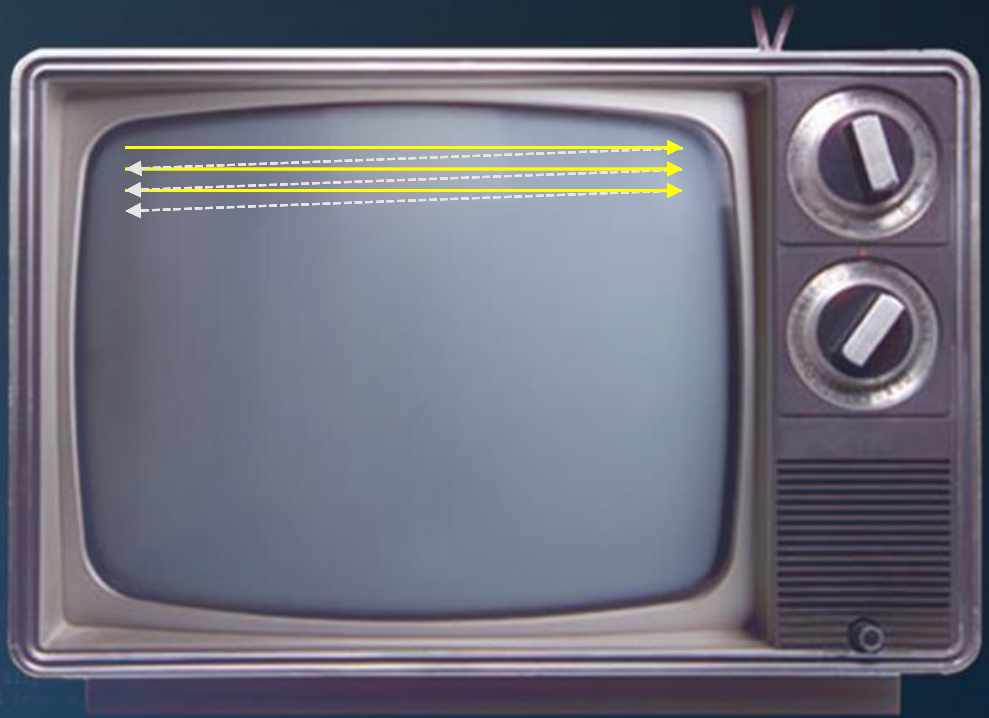


$$\pi = 4$$

$$a^2 + b^2 = \sqrt{a} + \sqrt{b}$$

Raster Displays

CRT – Cathode Ray Tube

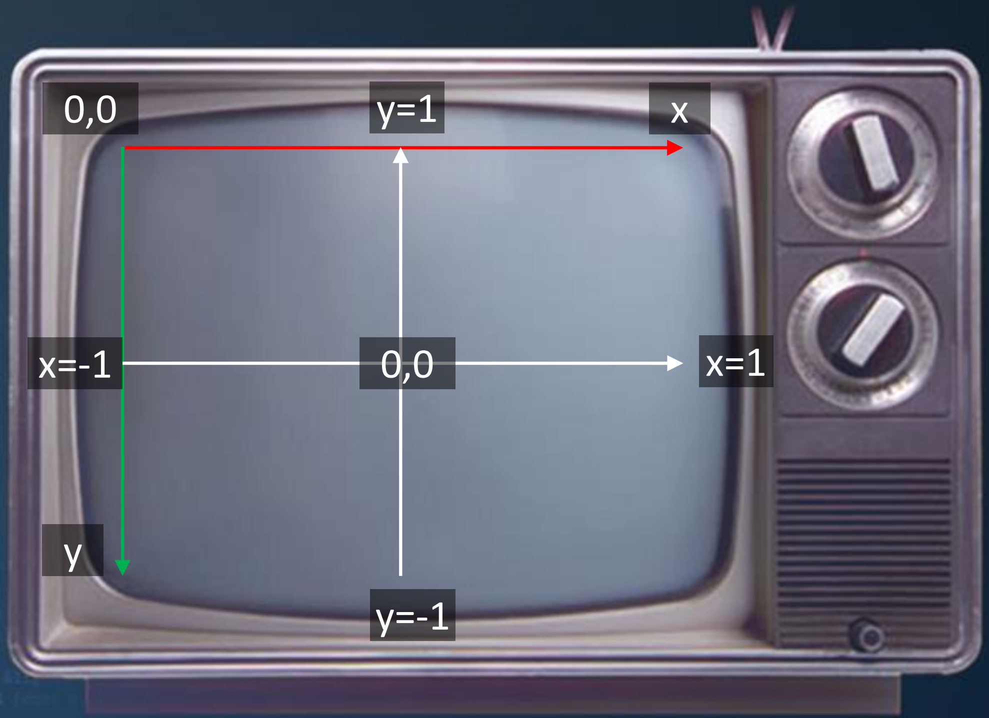


Physical implementation – origins

Electron beam zig-zagging over a fluorescent screen.

Raster Displays

CRT – Cathode Ray Tube



Physical implementation – consequences

- Origin in the top-left corner of the screen
- Axis system directly related to pixel count

Not the coordinate system we expected…

Raster Displays

Frame rate

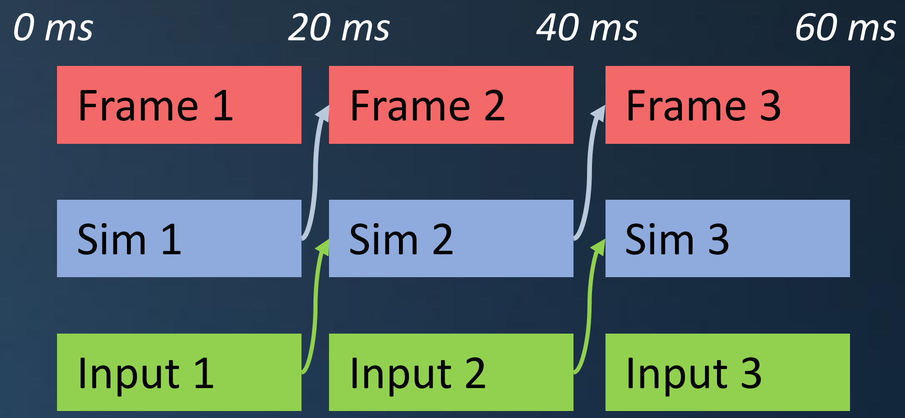
PAL: 25fps
NTSC: 30fps (actually: 29.97)
Typical laptop screen: 60Hz
High-end monitors: 120-240Hz
Cartoons: 12-15fps

Human eye:
‘Frame-less’
Not a raster.

How many fps / megapixels is ‘enough’?

Raster Displays

Frame rate



Raster Displays

Generating images

Rendering:
on a raster
“The process of generating an image from a 2D or 3D model by means of a computer program.” (Wikipedia)

Two main methods:

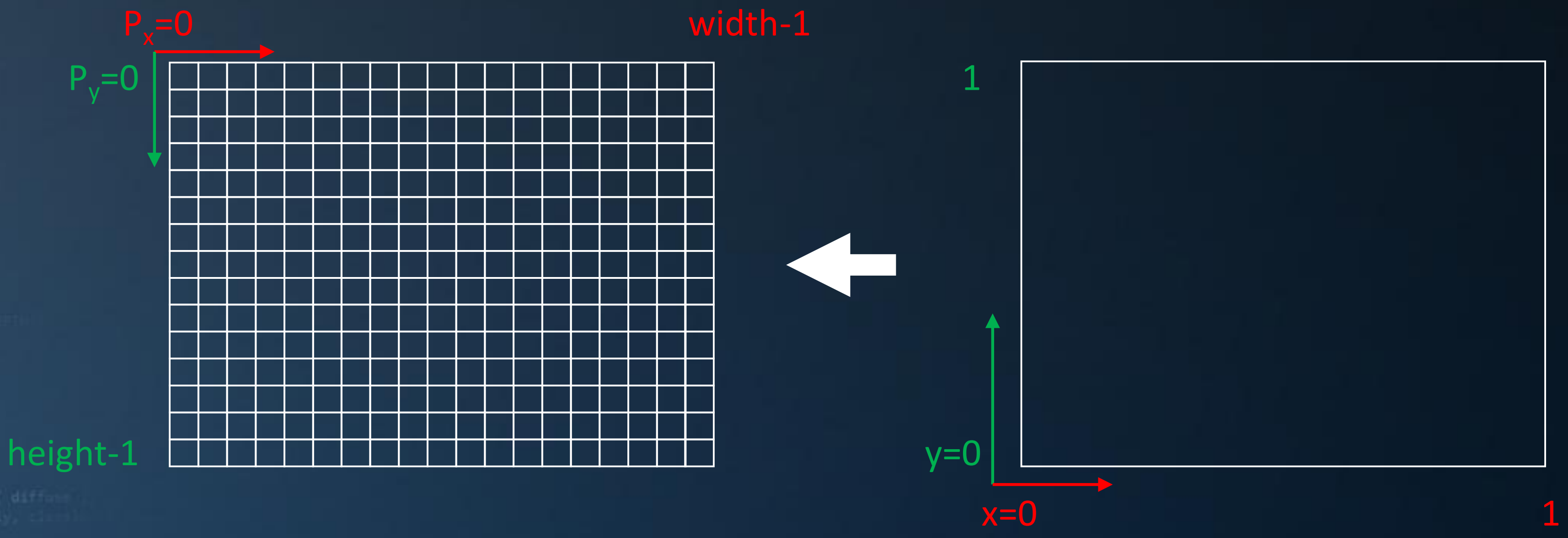
1. Ray tracing: for each pixel: what color do we assign to it?
2. Rasterization: for each triangle, which pixels does it affect?

Today's Agenda:

- The Raster Display
- Vector Math
- Colors

Vector Math

2D space



Conversion:

$P_x = x * \text{width}$

$P_y = (1-y) * \text{height}$

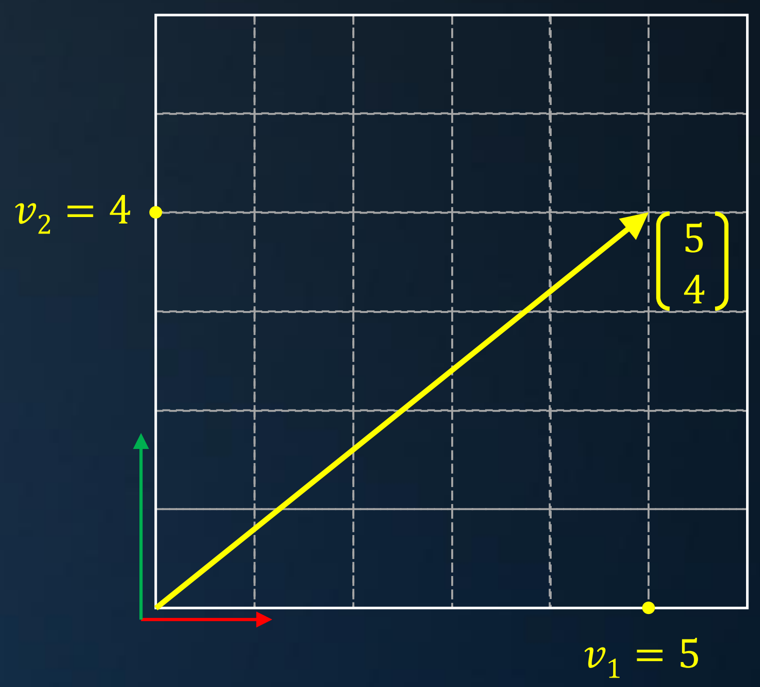
Vector Math

Vector Math

Vectors

In $\mathbb{R}^d$, a vector can be defined as an ordered *d*-tuple:

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \ldots \\ v_d \end{pmatrix}$$



A vector can also be defined by its *length* and *direction*.

The Euclidean length or *magnitude* of a vector is calculated using:

$$\|\vec{v}\|=\sqrt{v_1{}^2 + v_2{}^2 + \cdots + vd^2}$$

In 2D, this is similar to the Pythagorean theorem:

$$a^2 + b^2 = c^2$$

Vector Math

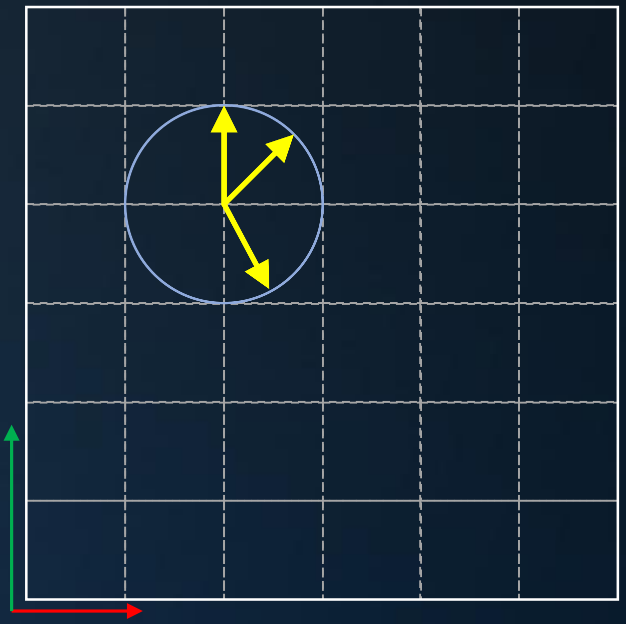
Vectors

A *unit vector* is a vector with length = 1:

$\|\vec{v}\| = 1$

A *null vector* is a vector with lenth = 0, e.g.:

in $\mathbb{R}^3$: $\vec{v} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$



A vector can be normalized by dividing it by its magnitude:

$$\vec{v}_{unit} = \frac{\vec{v}}{\|\vec{w}\|}$$

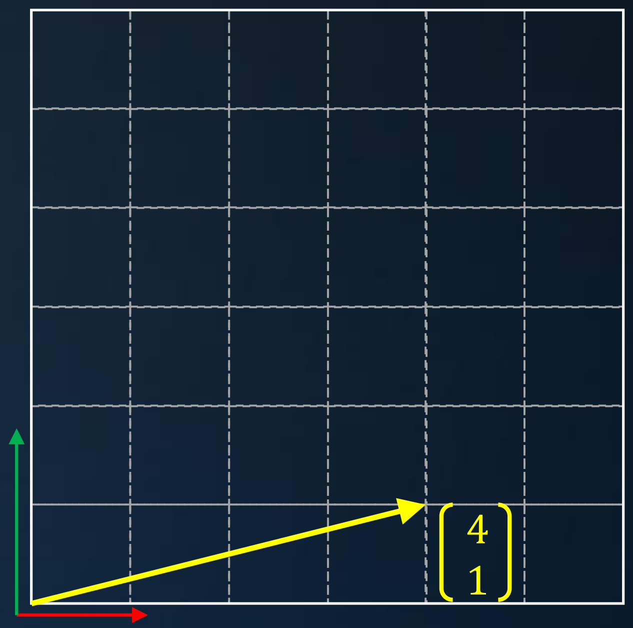
Can we normalize every vector?

Vector Math

Vectors

A 2D vector (x_v, y_v) can be seen as the point x_v, y_v in the Cartesian plane.

A 2D vector (x_v, y_v) can be seen as an *offset* from the *origin*.



Note:

Positions and vectors in $\mathbb{R}^3$ can be both represented by 3-tuples (x, y, z), but they are not the same!

Vector Math

Vectors

The sum of two vectors in $\mathbb{R}^d$,

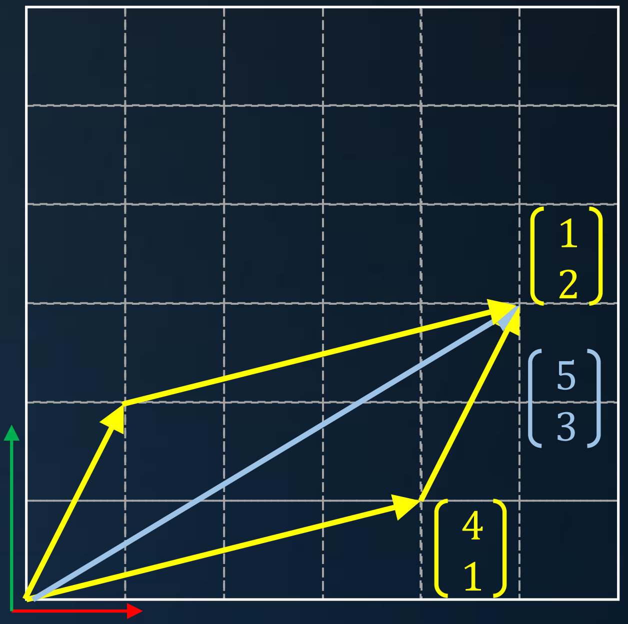
$\vec{v} = (v_1, v_2, .., v_d)$ and

$\vec{w} = (w_1, w_2, .., w_d)$

is defined as:

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

$(v_1 + w_1, v_2 + w_2, \dots, v_d + w_d)$



Example:

$(4,1) + (1,2) = (5,3)$

Vector subtraction is similarly defined.

Vector addition is *commutative* (as can be easily seen from the geometric interpretation):

$(4,1) + (1,2) = (5,3) = (1,2) + (4,1).$

Vector Math

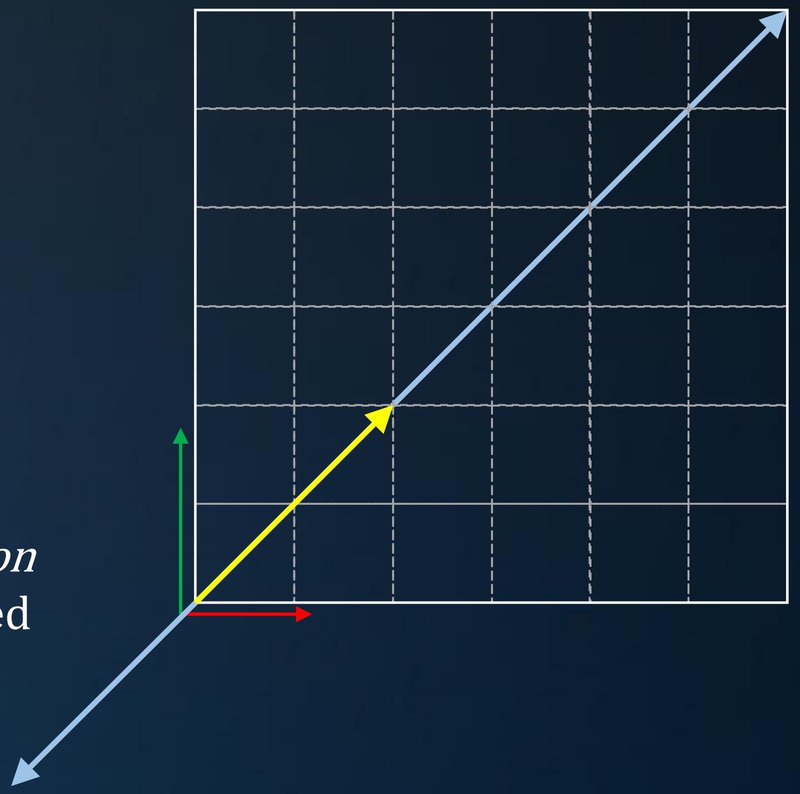
Vectors

The scalar multiple of a d-dimensional vector $\vec{v}$ is defined as:

$$\lambda\vec{v} = (\lambda v_1, \lambda v_2, \ldots, \lambda v_d)$$

Scalar multiplication can change the *length* of a vector.

It can also change the *direction* of the vector, which is reversed if $\lambda < 0$.



Two vectors $\vec{v}$ and $\vec{w}$ are parallel if one is a scalar multiple of the other, i.e.:

there is a λ such that $\vec{v} = \lambda\vec{w}$.

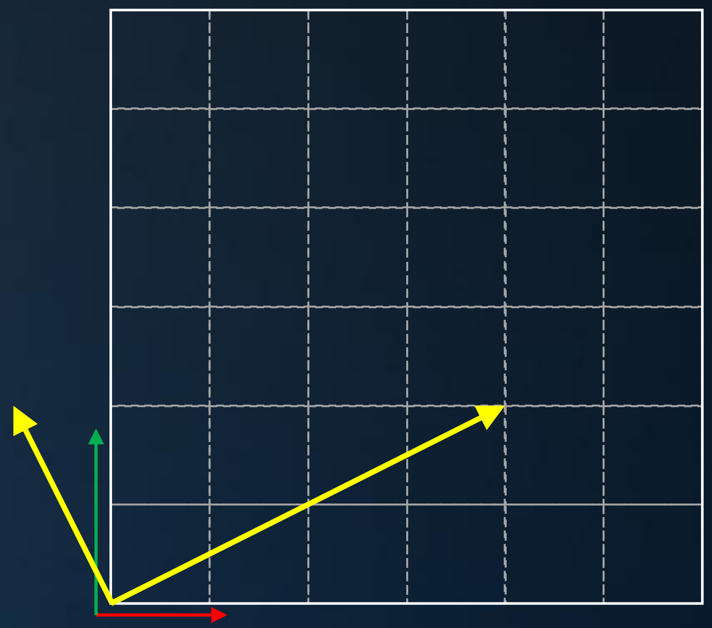
Vector Math

Vectors

Parallel vectors are called *linearly dependent.*

If they are not parallel, vectors are *linearly independent*.

A special case is when two vectors are perpendicular to each other; in this case, each vector is the *normal vector* of the other.



In $\mathbb{R}^2$, we can easily create a normal vector for (v_x, v_y):

$$\vec{n} = (-v_y, v_x)$$

Question: does this also work in $\mathbb{R}^3$?

Vector Math

Bases

We can use two linearly independent vectors to produce any vector:

$$\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$$

This doesn’t just work for perpendicular vectors.

Vector Math

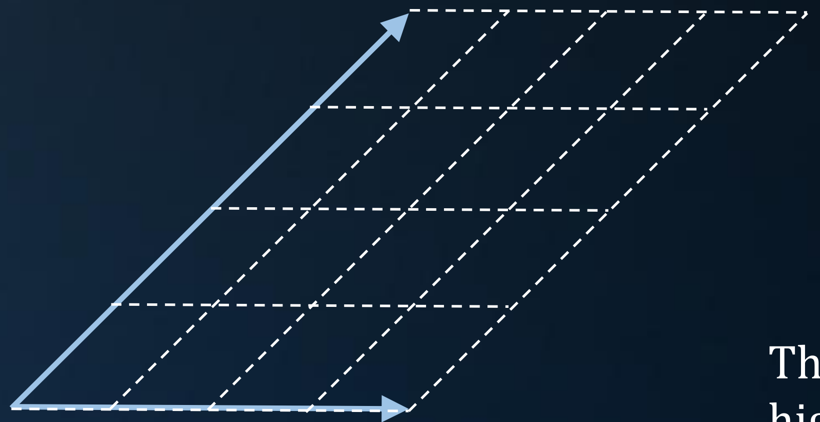
Bases

We can use two linearly independent vectors to produce any vector:

$$\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$$

This doesn’t just work for perpendicular vectors.

Any pair of linearly independent vectors form a *2D basis*.



This extends naturally to higher dimensions.

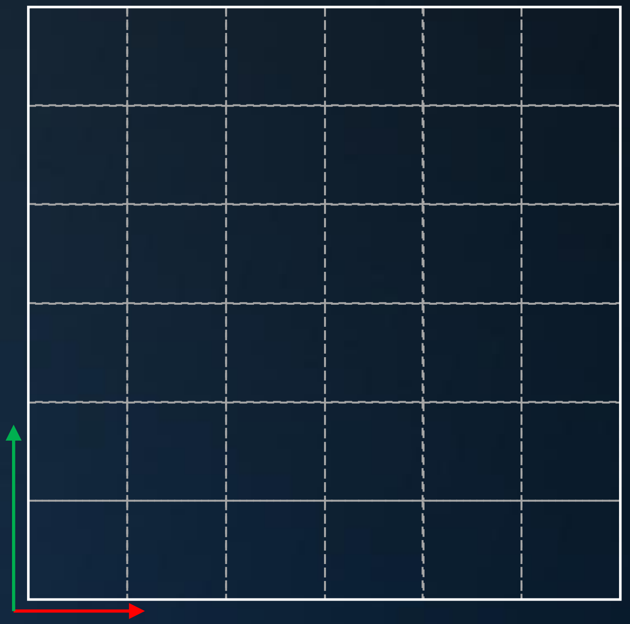
Vector Math

Bases

“Any pair of linearly independent vectors form a 2D basis”:

The Cartesian coordinate system is an example of this.

In this case the vectors (1,0) and (0,1) form an *orthonormal* basis:



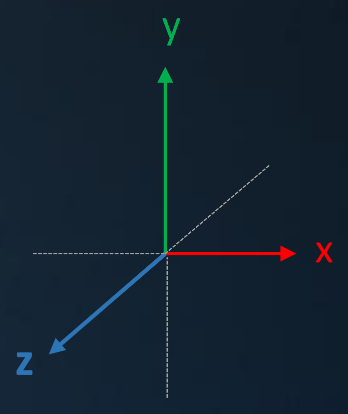
1. The vectors are orthogonal to each other;
2. The vectors are unit vectors.

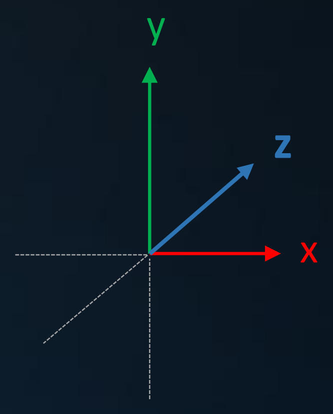
Vector Math

Bases

A coordinate system can be *left handed* or *right handed.*

Note that this only affects the interpretation of the vectors; the vectors themselves are the same in each case.





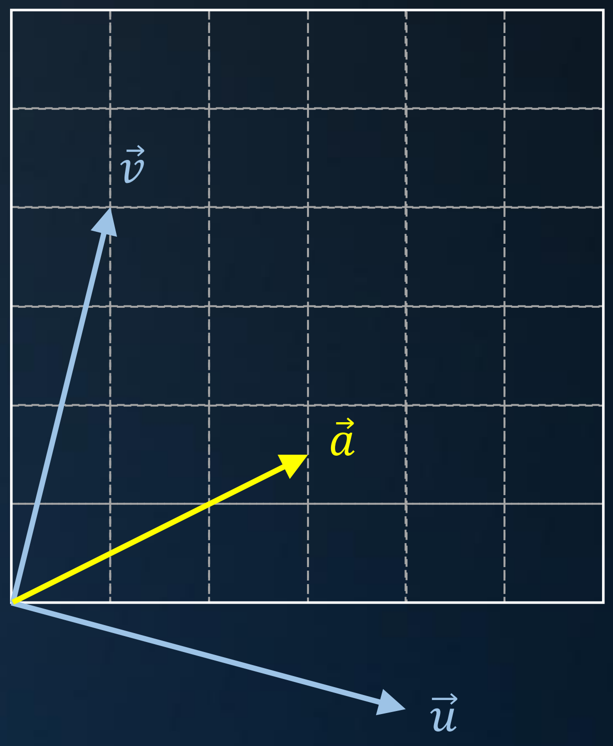
Vector Math

Dot product

Given vectors $\vec{a}$, $\vec{u}$ and $\vec{v}$, we know that:

$$\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$$

We can determine λ_1 and λ_2 using the *dot product**.



The dot product of vector $\vec{v}$ and $\vec{w}$ is defined as:

$$\vec{v} \cdot \vec{w} = v_1 w_1 + v_2 w_2 + \cdots + v_d w_d$$

or

$$\vec{v} \cdot \vec{w} = \sum_{i=0}^{d} v_i w_i$$

*: AKA *inner product* or *scalar product*

Vector Math

Dot product

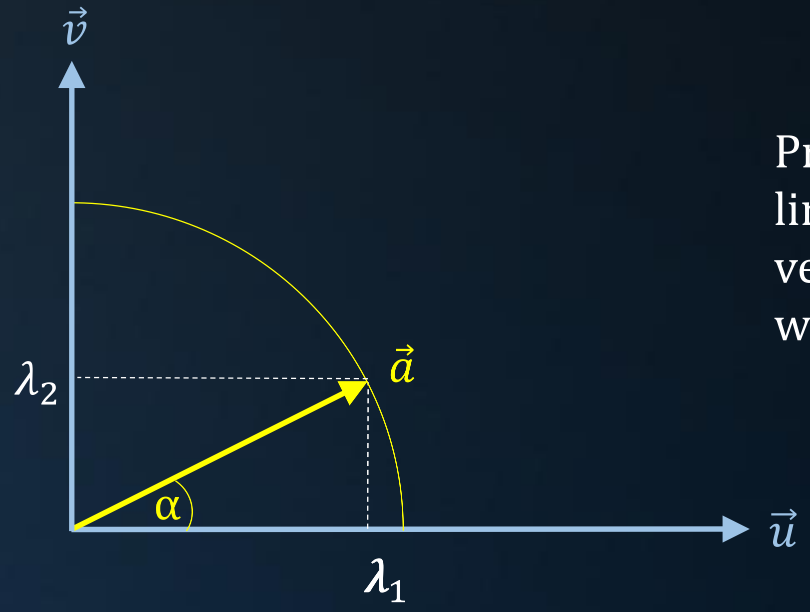
The dot product *projects* one vector on another.

If $\vec{a}$ and $\vec{u}$ are unit vectors, we can calculate the angle between them using the dot product:

$$\lambda = \cos \propto = \vec{u} \cdot \vec{a}$$

or, if they are not normalized:

$$\cos \propto = \frac{\vec{u} \cdot \vec{a}}{\|\vec{u}\| \ \|\vec{a}\|}$$



Projecting a vector on two linearly independent vectors yields a coordinate within the 2D basis.

This works regardless of the direction and scale of $\vec{u}$ and $\vec{v}$, and also in $\mathbb{R}^3$.

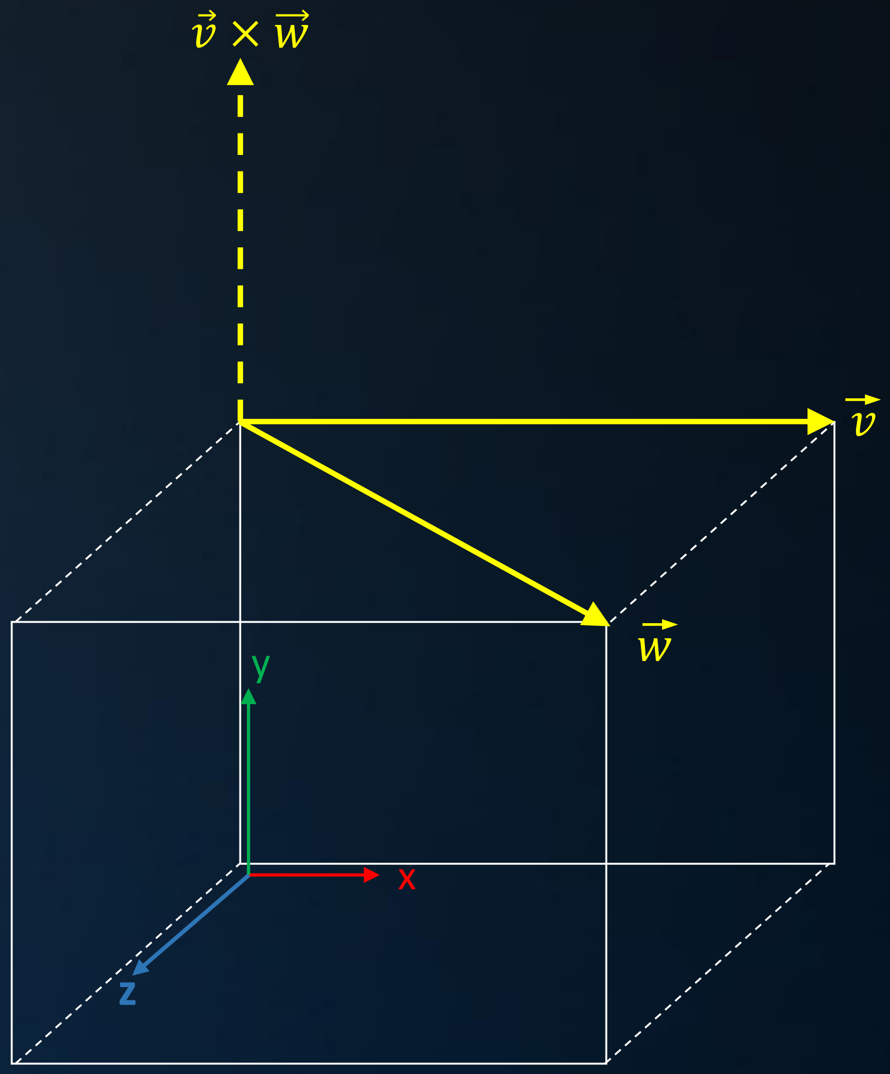
2D Transforms

Cross product

The cross product can be used to calculate a vector perpendicular to a 2D basis formed by 2 vectors. It is defined as:

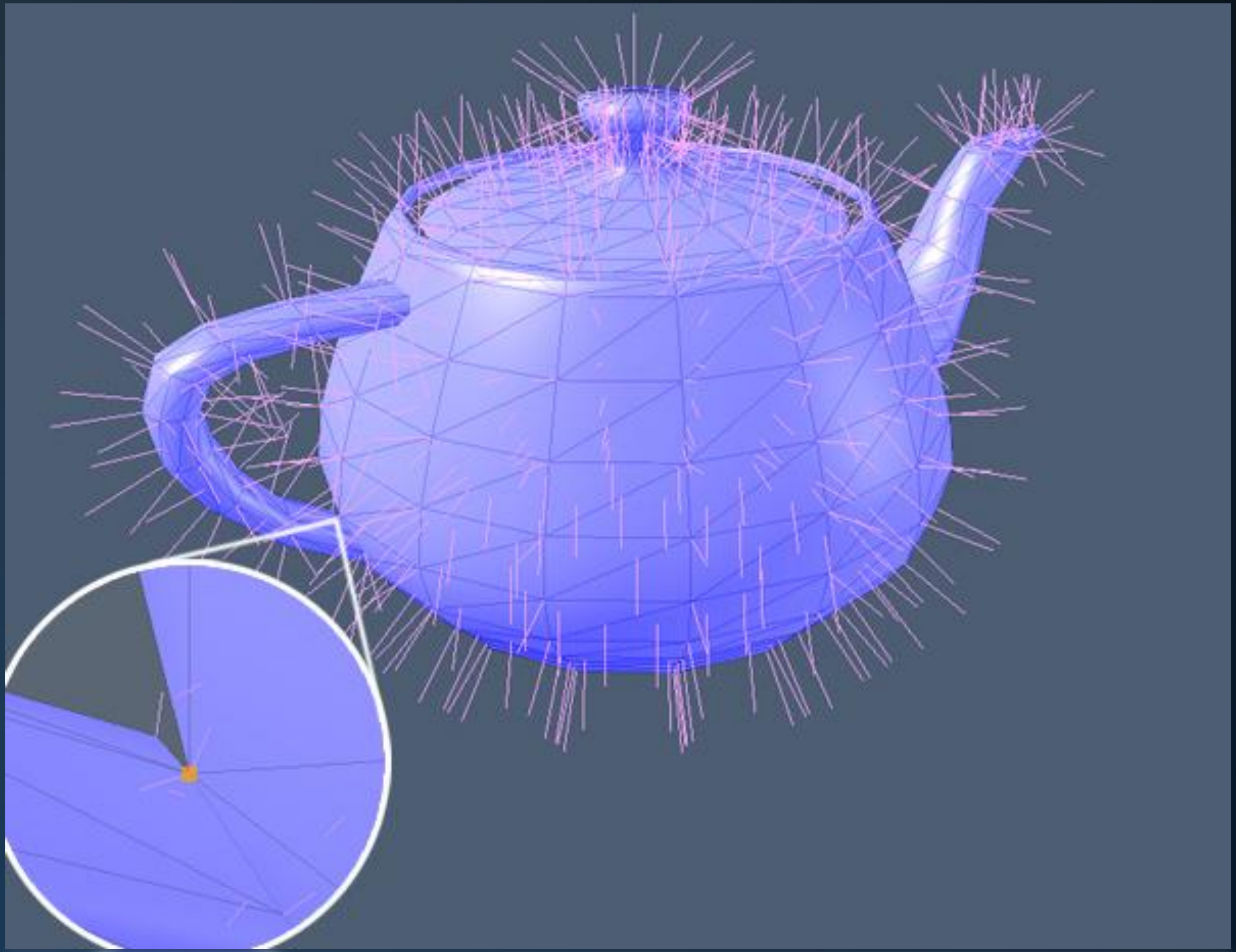
$$\vec{v} \times \vec{w} = \begin{pmatrix} v_2 w_3 - v_3 w_2 \\ v_3 w_1 - v_1 w_3 \\ v_1 w_2 - v_2 w_1 \end{pmatrix}$$

Note:
The cross product is only defined in $\mathbb{R}^3$.

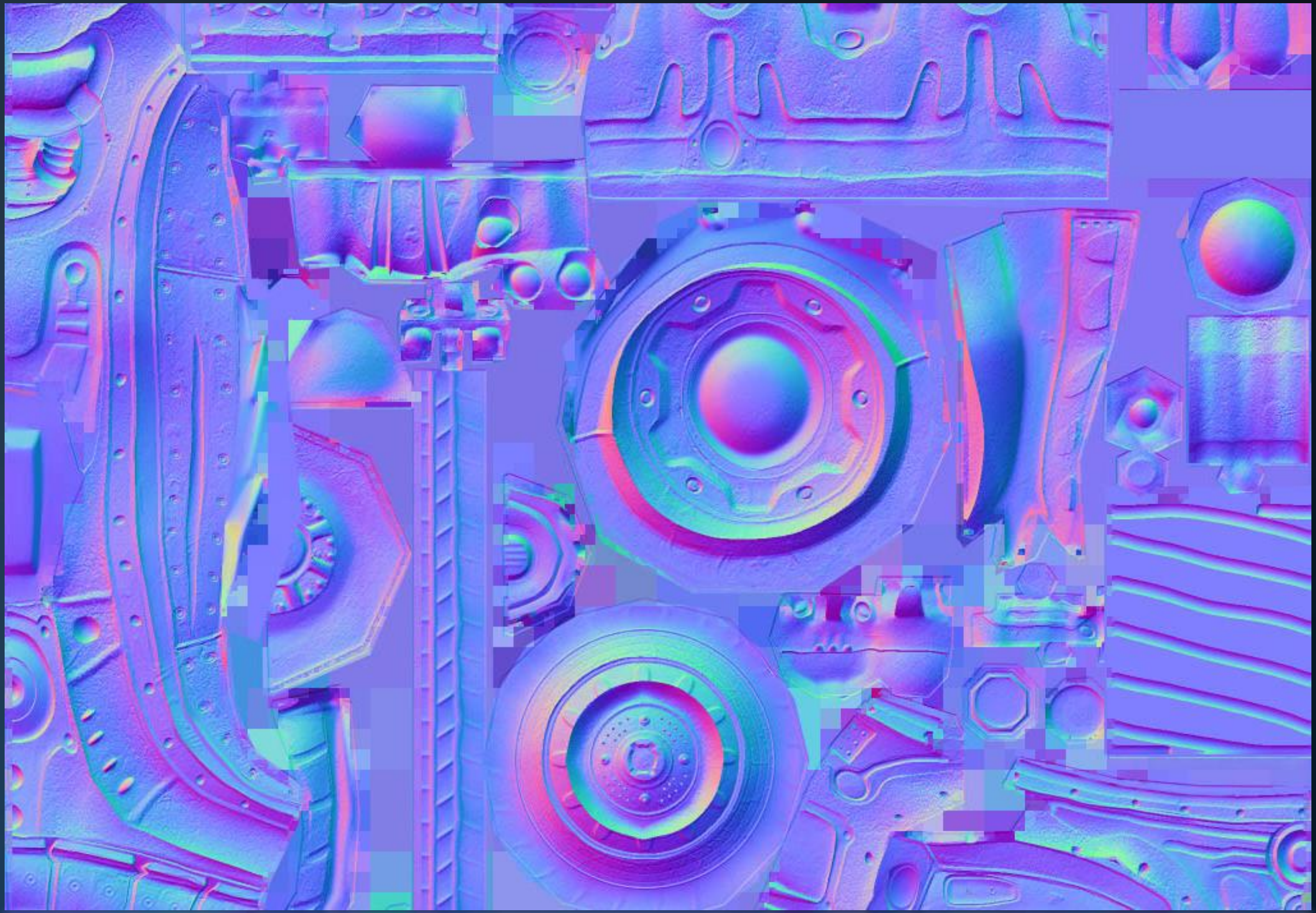


2D Transforms

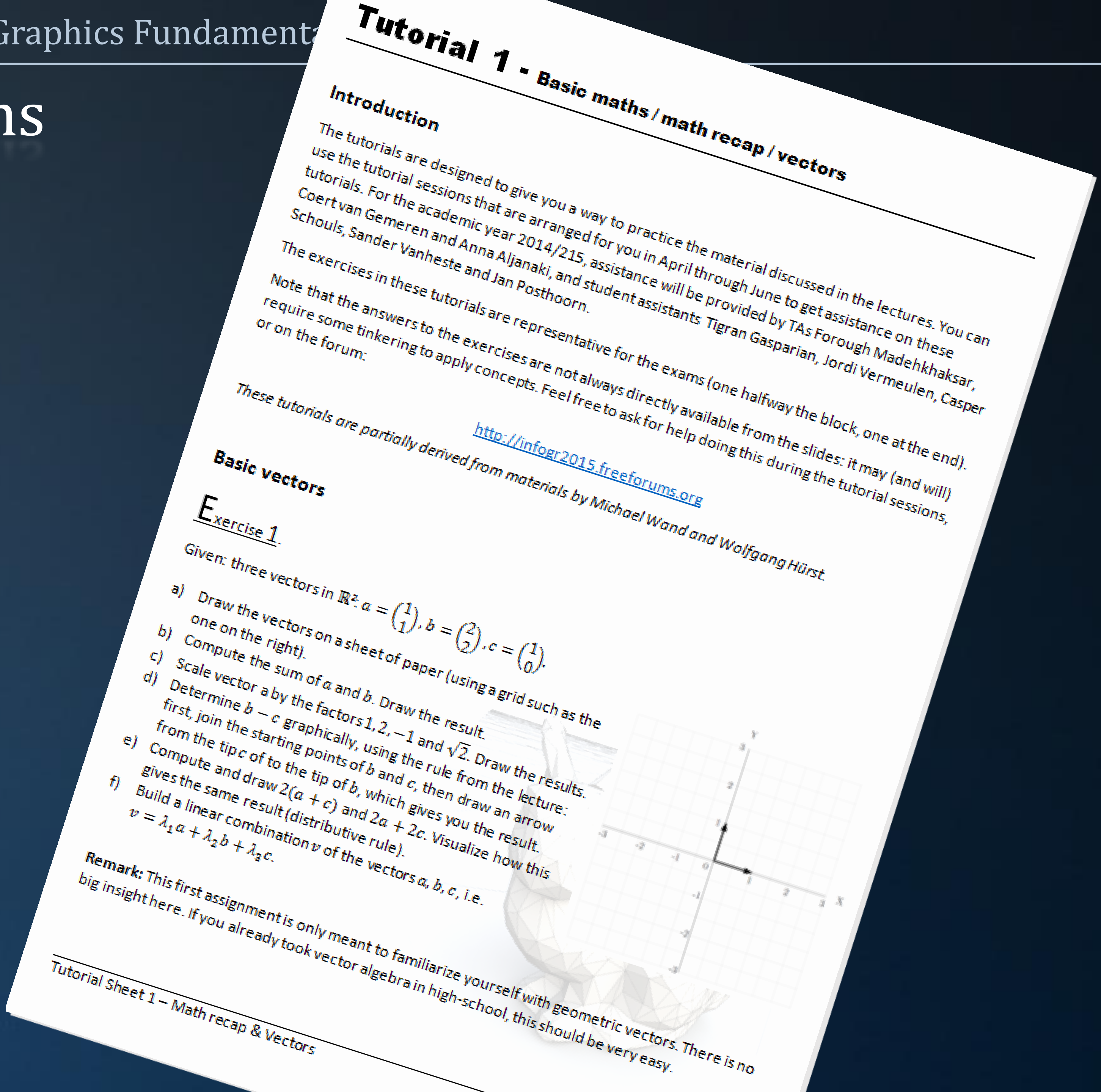
2D Transforms



2D Transforms



2D Transforms



Tutorial 1 - Basic maths / math recap / vectors

Introduction

The tutorials are designed to give you a way to practice the material discussed in the lectures. You can use the tutorial sessions that are arranged for you in April through June to get assistance on these tutorials. For the academic year 2014/215, assistance will be provided by TAs Forough Madehkhaksar, Coert van Gemeren and Anna Aljanaki, and student assistants Tigran Gasparian, Jordi Vermeulen, Casper Schouls, Sander Vanheste and Jan Posthoorn.

The exercises in these tutorials are representative for the exams (one halfway the block, one at the end).

Note that the answers to the exercises are not always directly available from the slides: it may (and will) require some tinkering to apply concepts. Feel free to ask for help doing this during the tutorial sessions, or on the forum:

http://infogr2015.freeforums.org

These tutorials are partially derived from materials by Michael Wand and Wolfgang Hürst.

Basic vectors

Exercise 1.

Given: three vectors in $\mathbb{R}^2$: $a = \binom{1}{1}$, $b = \binom{2}{2}$, $c = \binom{1}{0}$.

a) Draw the vectors on a sheet of paper (using a grid such as the one on the right).
b) Compute the sum of a and b. Draw the result.
c) Scale vector a by the factors $1, 2, -1$ and $\sqrt{2}$. Draw the results.
d) Determine $b - c$ graphically, using the rule from the lecture: first, join the starting points of b and c, then draw an arrow from the tip c of to the tip of b, which gives you the result.
e) Compute and draw $2(a + c)$ and $2a + 2c$. Visualize how this gives the same result (distributive rule).
f) Build a linear combination v of the vectors a, b, c, i.e. $v = \lambda_1 a + \lambda_2 b + \lambda_3 c$.

Remark: This first assignment is only meant to familiarize yourself with geometric vectors. There is no big insight here. If you already took vector algebra in high-school, this should be very easy.

Tutorial Sheet 1 – Math recap & Vectors

Today’s Agenda:

- The Raster Display
- Vector Math
- Colors

Colors

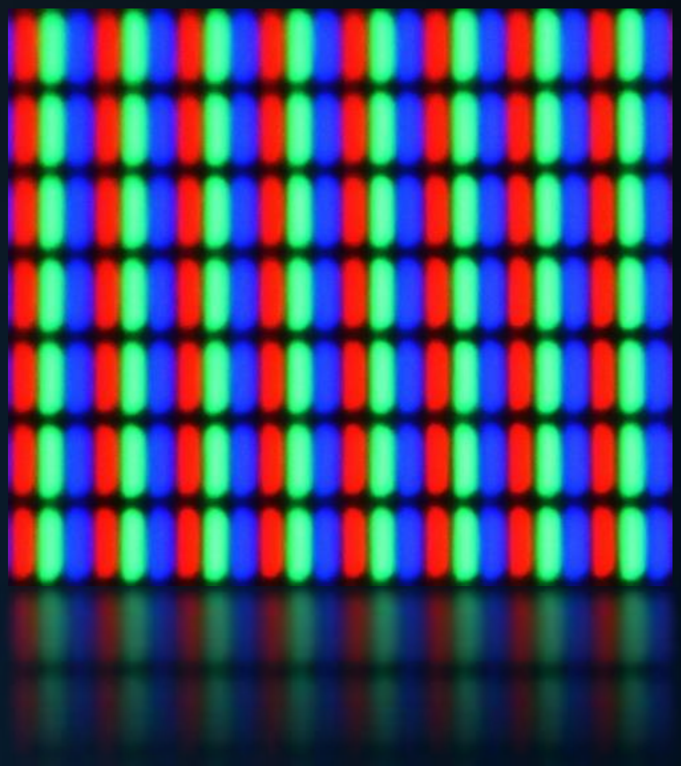
Color representation

Computer screens emit light in three colors: red, green and blue.

By additively mixing these, we can produce most colors: from black (red, green and blue turned off) to white (red, green and blue at full brightness).

In computer graphics, colors are stored in discrete form. This has implications for:

- Color resolution (i.e., number of unique values per component);
- Maximum brightness (i.e., range of component values).



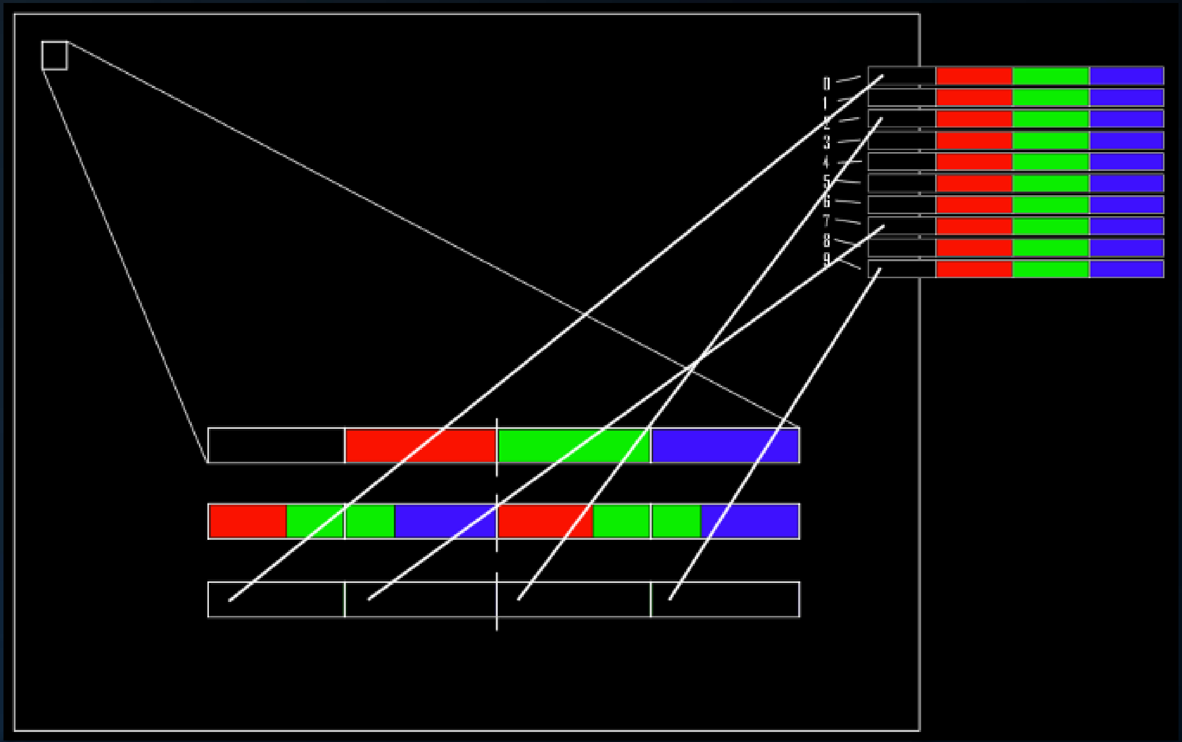
Colors

Color representation

The most common color representation is 32-bit ARGB, which stores red, green and blue as 8 bit values (0..255).

Alternatively, we can use 16 bit for one pixel (RGB 565),

or a color palette. In that case, one byte is used per pixel, but only 256 unique colors can be used for the image.



Colors

Color representation

Colors

Color representation

Colors

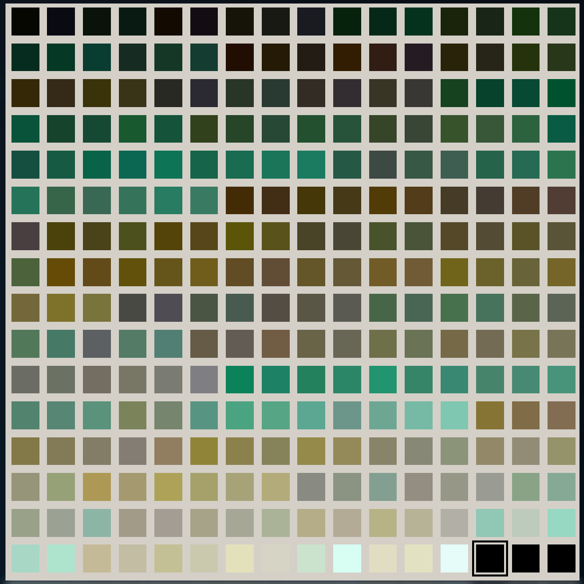
Color representation

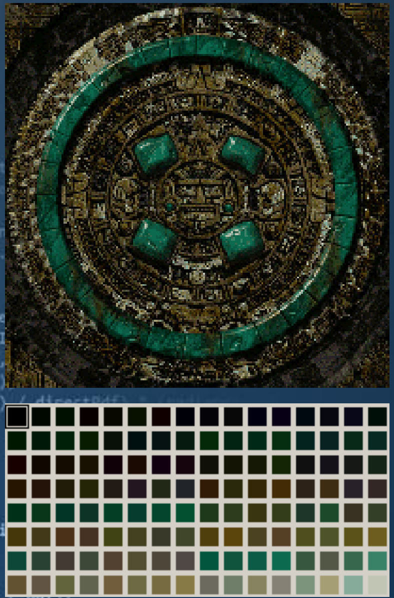
Colors

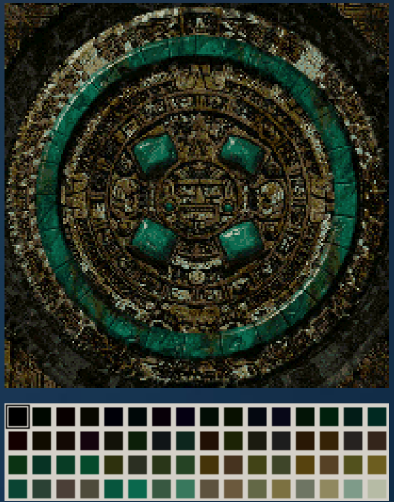
Color representation

Textures can typically safely be stored as palletized images.

Using a smaller palette will result in smaller compressed files.







Colors

Color representation

Using a fixed range (0:0:0 … 255:255:255) places a cap on the maximum brightness that can be represented:

- A white sheet of paper: (255,255,255)
- A bright sky: (255,255,255)

The difference becomes apparent when we look at the sky and the sheet of paper through sunglasses.

(or, when the sky is reflected in murky water)

Colors

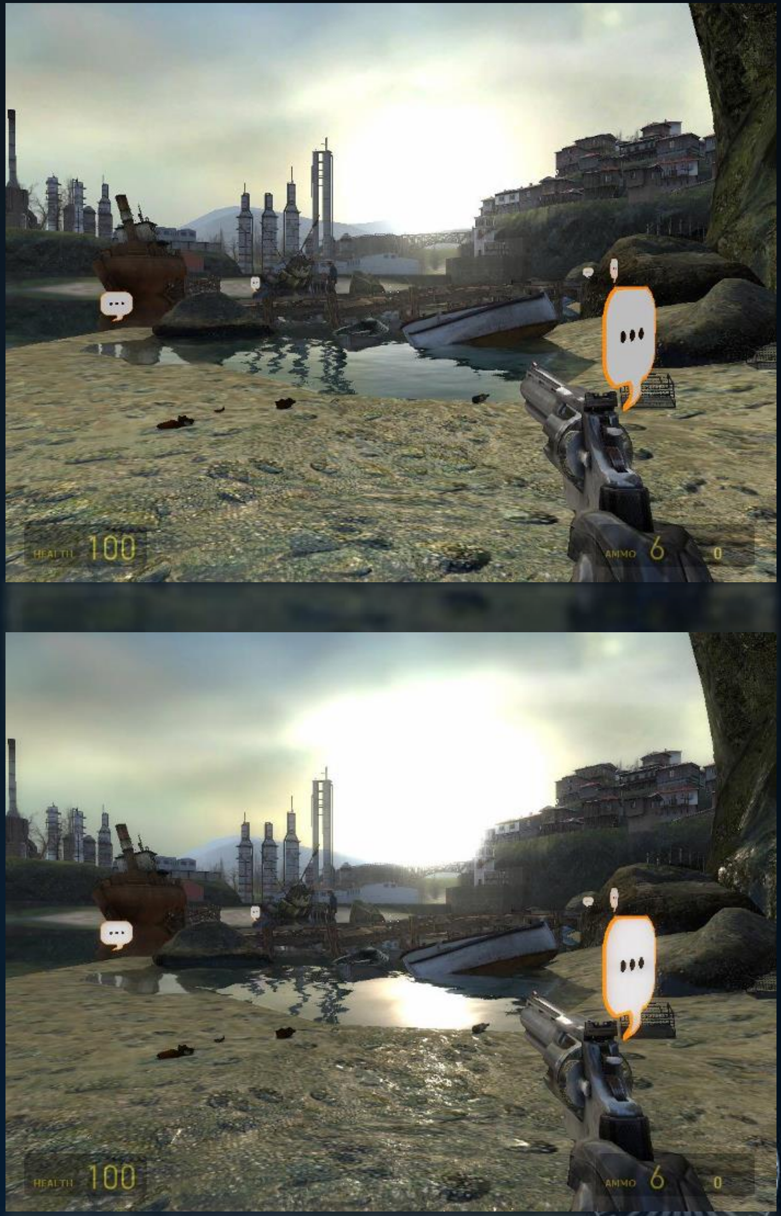
Color representation

For realistic rendering, it is important to use an internal color representation with a much greater range than 0..255 per color component.

HDR: High Dynamic Range;

We store one float value per color component.

Including alpha, this requires 128bit per pixel.



Today’s Agenda:

- The Raster Display
- Vector Math
- Colors

INFOGR – Computer Graphics

Jacco Bikker - April-July 2015 - Lecture 2: “Graphics Fundamentals”

END of “Graphics Fundamentals”

next lecture: “Geometry”