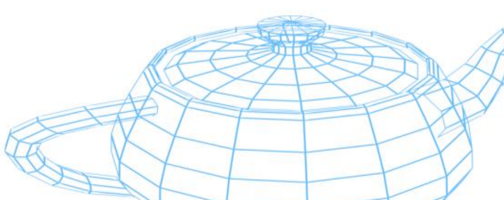
Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

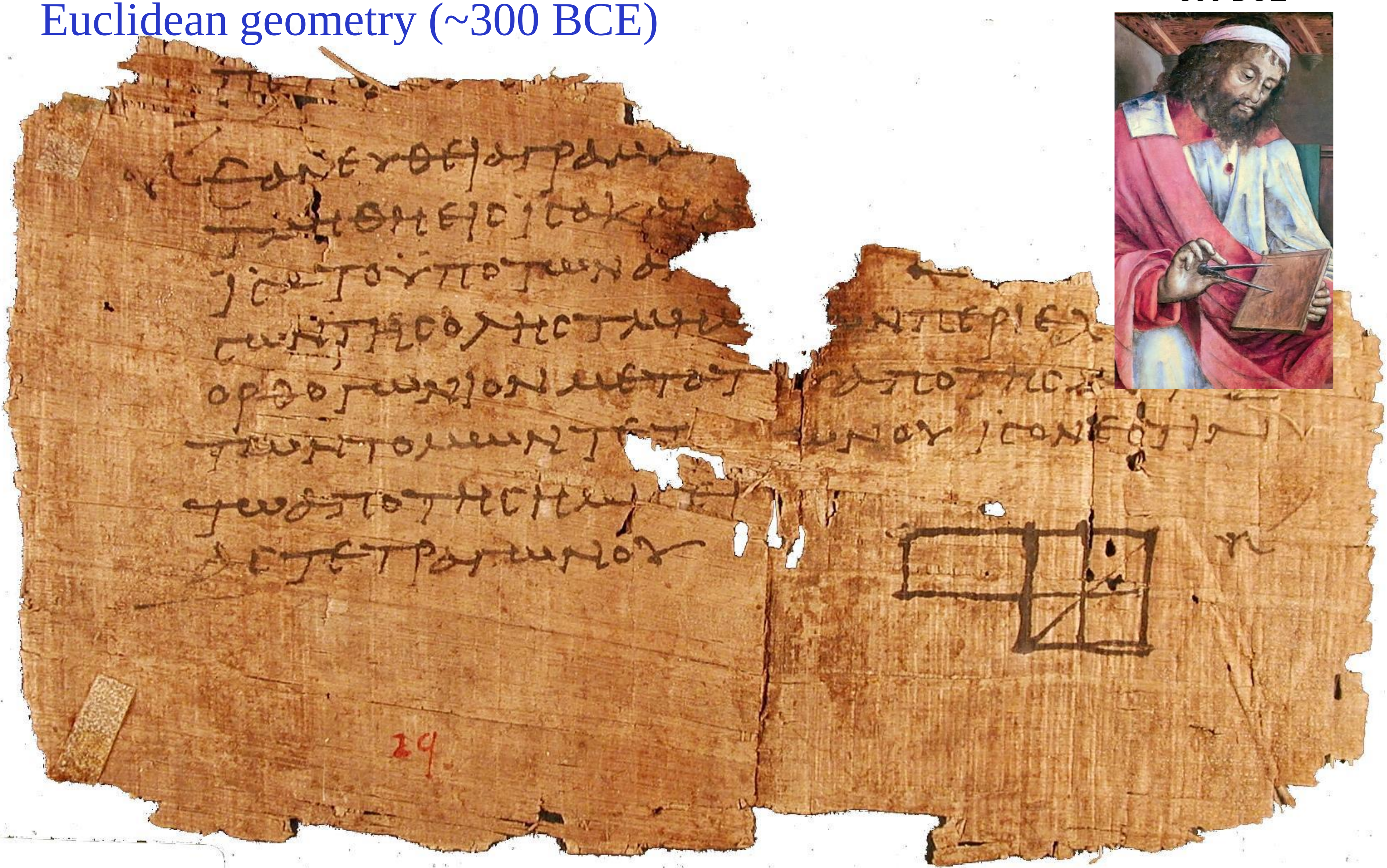
Computer Graphics

Mathematics for Computer Graphics (1)

Achille Peternier, adjunct professor

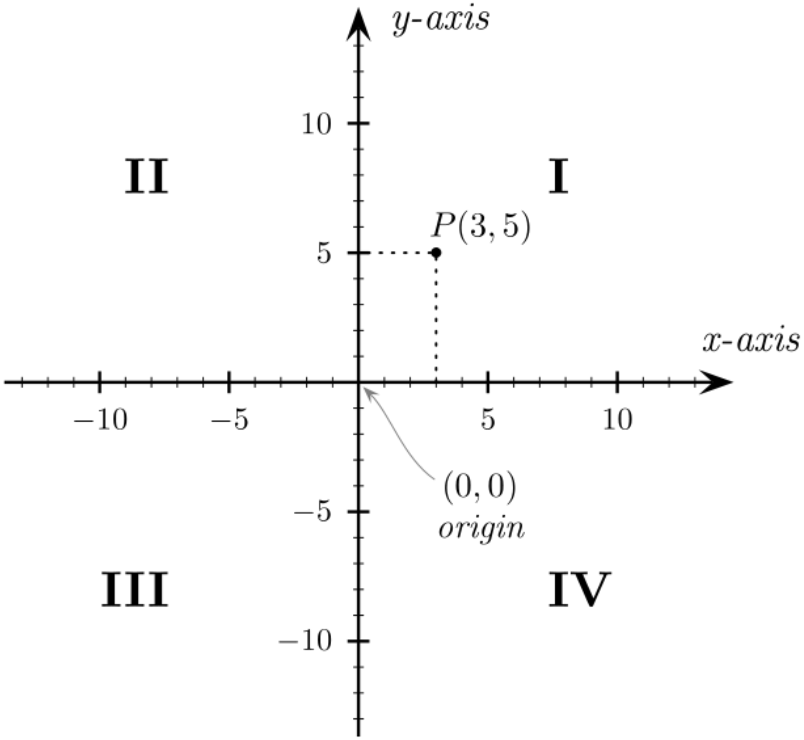


Euclidean geometry (~300 BCE)



Euclid of Alexandria
~300 BCE

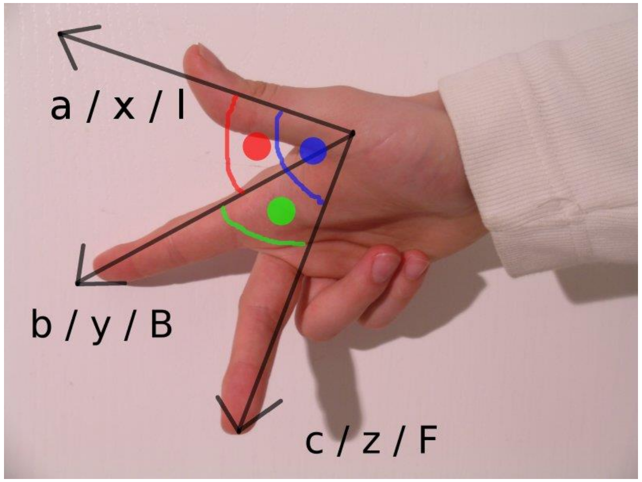
Coordinate systems (1637)

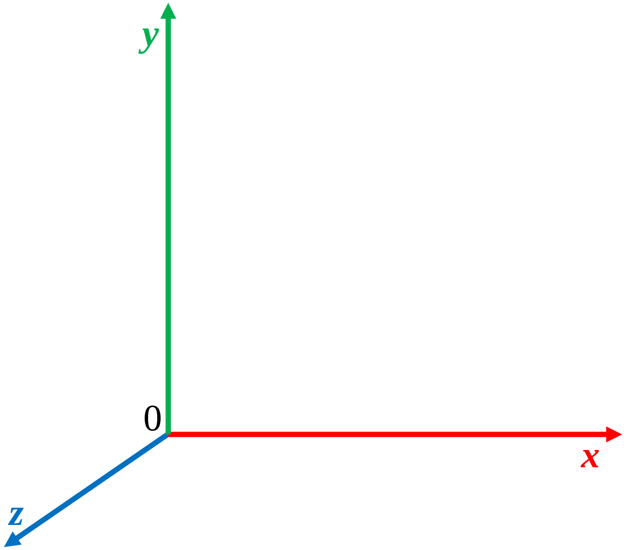


René Descartes
1596 - 1650

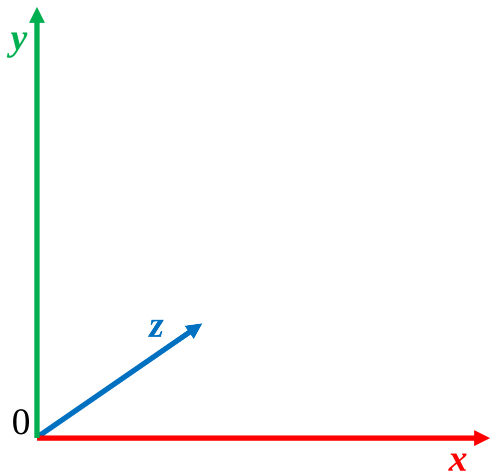
Coordinate systems

- Right-hand rule.

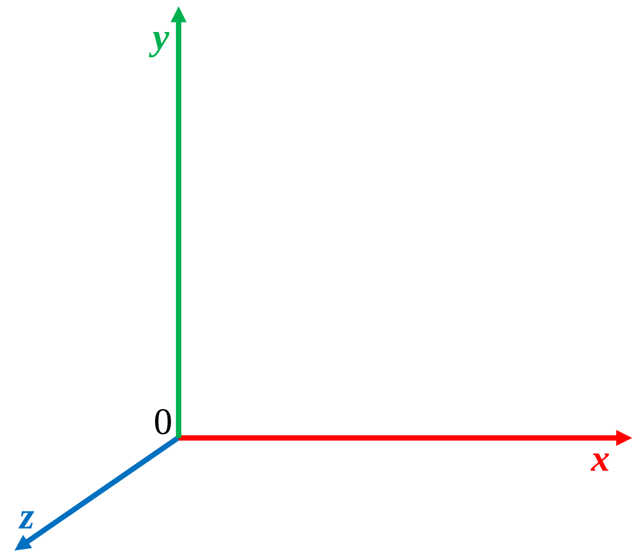




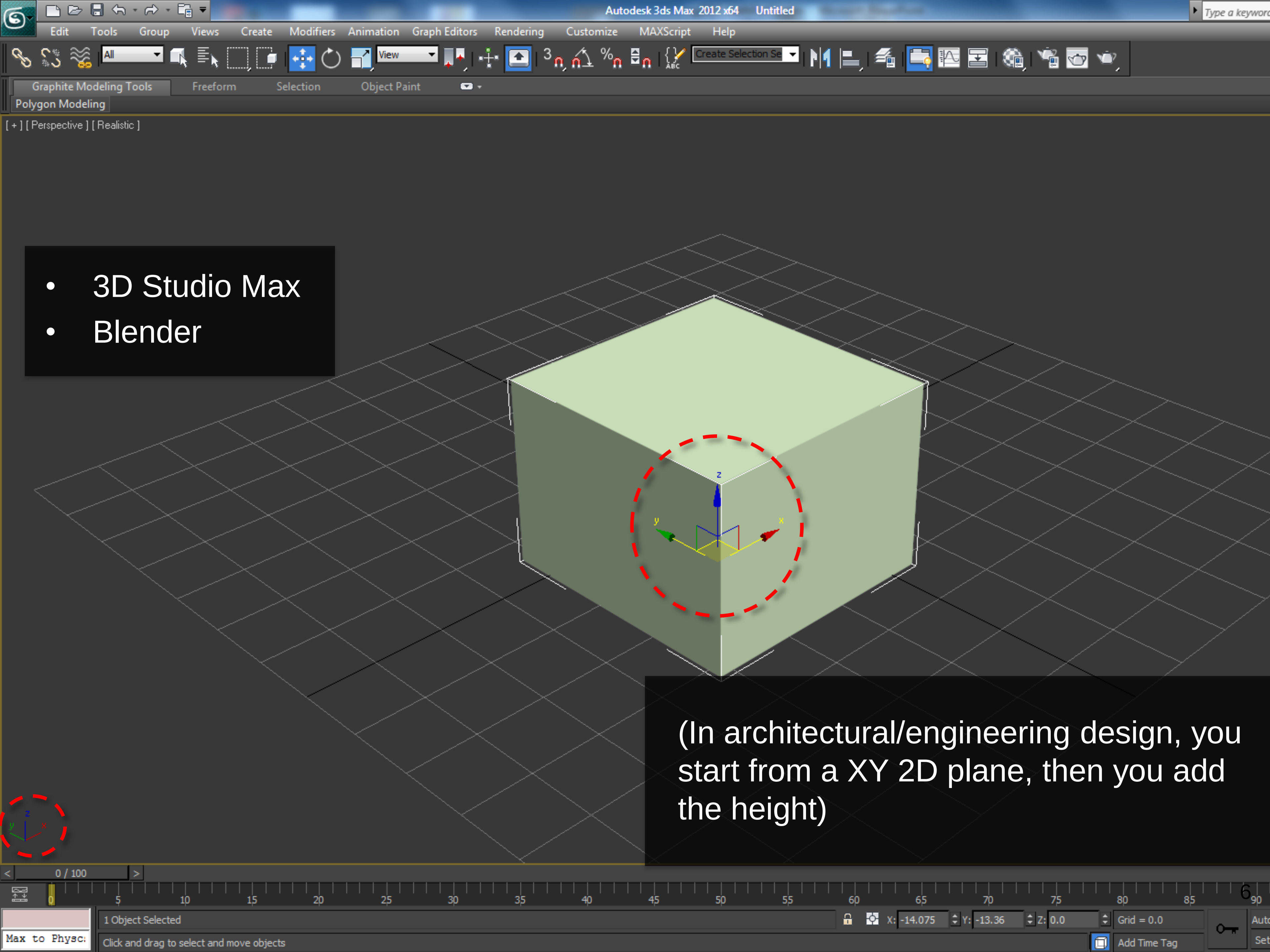
Coordinate systems



(left-hand rule)



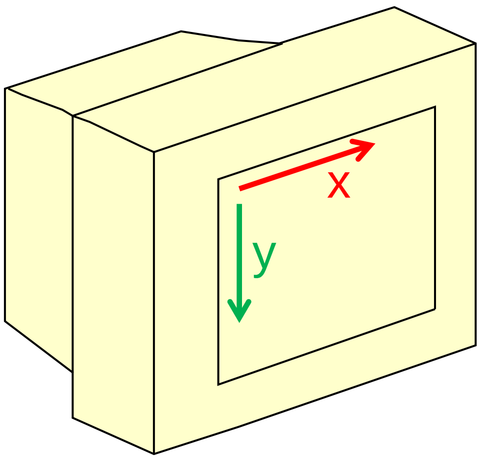
- 3D Studio Max
- Blender



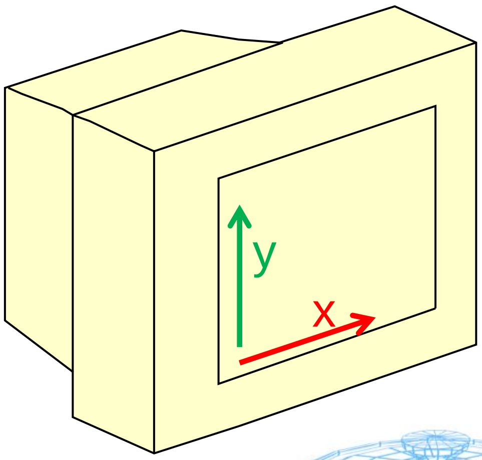
(In architectural/engineering design, you start from a XY 2D plane, then you add the height)

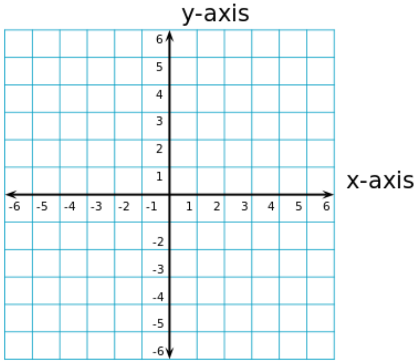
Coordinate systems

- Screen coordinates:
  - Origin located at the top-left corner:
    - Windows, X11.



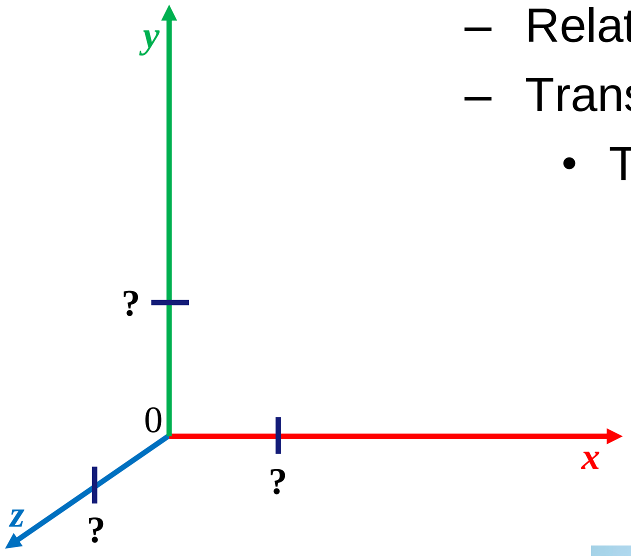
  - Origin located at the bottom-left corner:
    - MacOS UI, OpenGL.





Units

- Generic units without explicit meaning:
  - Relative to each other.
  - Translated into pixels only at the end of the rendering pipeline:
    - To match different screen resolutions and aspect ratios.

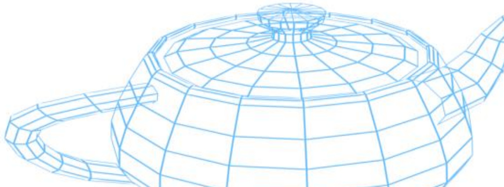


Point

- Defines a location.

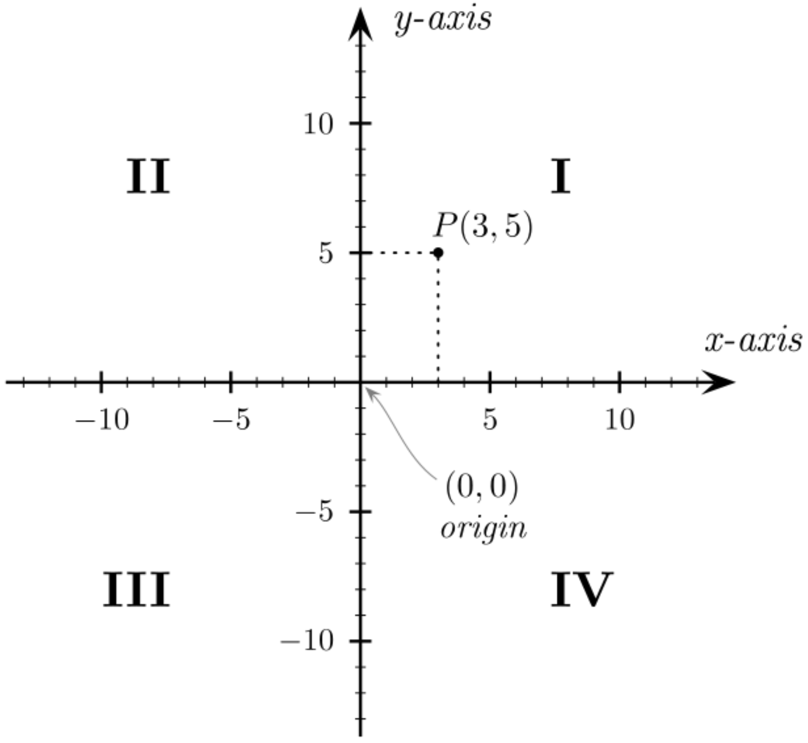
- Abstract entity:
  - no length.
  - no thickness.
  - no direction.

- A point usually specifies one of the vertices
of a 3D primitive or the position of a light source.

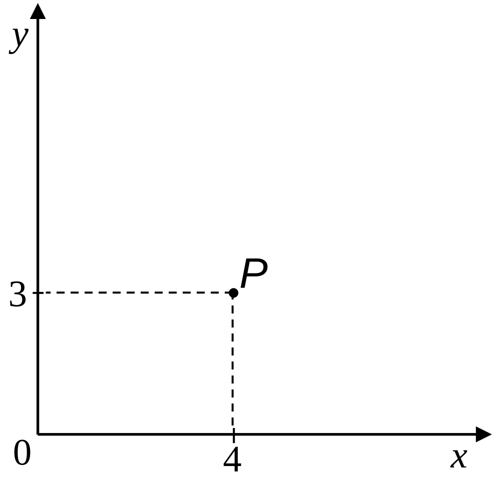


Point

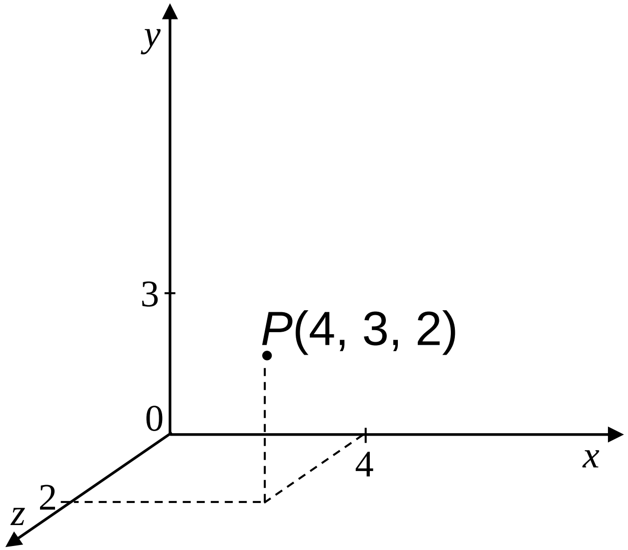
- $P(3, 5)$



Point



$P(4, 3)$



(4, 3)

$$\begin{bmatrix}4\\3\end{bmatrix}$$

$$\begin{bmatrix}4 & 3\end{bmatrix}^T$$

(4, 3, 2)

$$\begin{bmatrix}4\\3\\2\end{bmatrix}$$

$$\begin{bmatrix}4 & 3 & 2\end{bmatrix}^T$$

Point

- Standard notation: (x, y, z)
- Row matrix notation: $\begin{bmatrix} x & y & z \end{bmatrix}$ or $\begin{pmatrix} x & y & z \end{pmatrix}$
- Column matrix notation: $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ or $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ OpenGL
- Transposed notation: $\begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}}$

Point

$$(4, 3) \neq \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$(4, 3) \neq [4 \;\; 3]^T$$

$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = [4 \;\; 3]^T$$

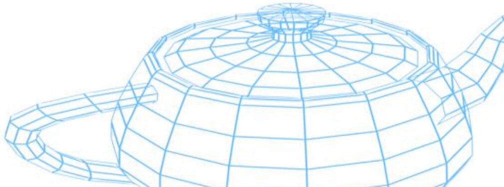
$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

$$(4, 3, 2) \neq \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix}$$

$$(4, 3, 2) \neq [4 \;\; 3 \;\; 2]^T$$

$$\begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix}^T = [4 \;\; 3 \;\; 2]$$

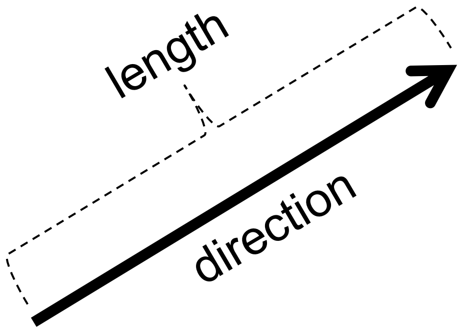
$$\begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \end{pmatrix}$$



Vector

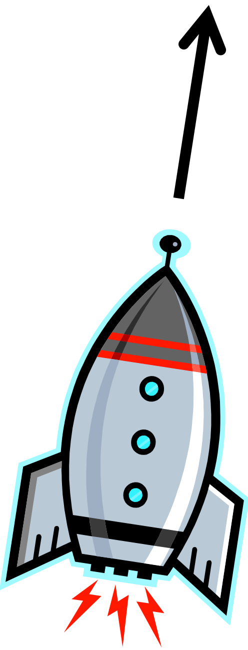
- Specifies a displacement:
  - Direction.
  - Length (magnitude).

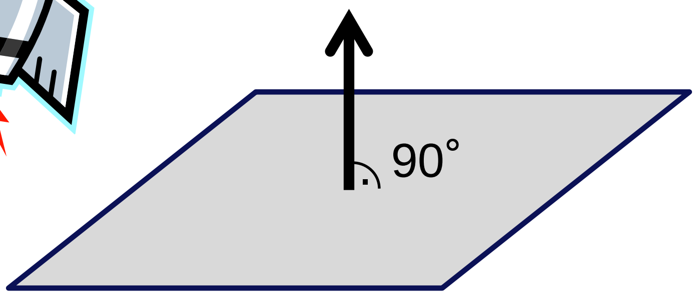
- <u>No</u> position.



Vector

- Typical usage:
  - Light direction and intensity.
  - Object displacement.
  - Surface orientation (normal).
  - Physical properties (e.g., gravity, acceleration).
  - Wind direction.
  - …





Notation

- Conventions:
  - bold lowercase letter for column and row matrices, using the first letters of the alphabet for known elements:

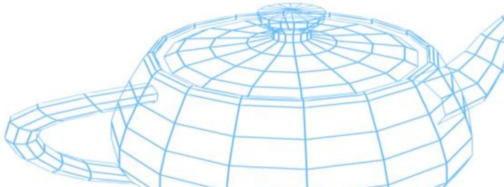
$$\mathbf{a} = \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix} \qquad \mathbf{b} = [1 \;\; 2 \;\; 3] \qquad \mathbf{c} = [0 \;\; -1 \;\; 2]^T$$

  - …and letters from the end of the alphabet when elements are variables:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \mathbf{r} = [r_0 \;\; r_1 \;\; r_2] \qquad \mathbf{t} = [t_1 \;\; t_2 \;\; t_3]^T$$

  - Bar or arrow instead of bold:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \vec{b} = [1 \;\; 2 \;\; 3]$$



Zero vector

- Defined by **0** (bold zero) or $\vec{0}$:

$$\mathbf{0} = [0 \;\; 0 \;\; \dots \;\; 0]$$

- …such as:

$$\mathbf{a} + \mathbf{0} = \mathbf{a} \qquad\qquad \mathbf{b} - \mathbf{0} = \mathbf{b}$$

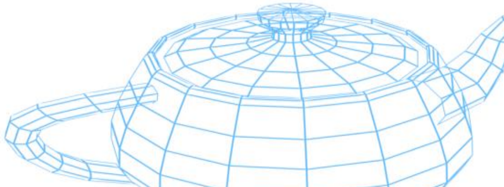
Vector operations

- Addition:

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a} = [a_1 + b_1 \quad a_2 + b_2 \quad \dots \quad a_n + b_n]$$

- Subtraction:

$$\mathbf{a} - \mathbf{b} = [a_1 - b_1 \quad a_2 - b_2 \quad \dots \quad a_n - b_n] \qquad \mathbf{a} - \mathbf{b} \neq \mathbf{b} - \mathbf{a} \qquad -\mathbf{a} = [-a_1 \quad -a_2 \quad \dots \quad -a_n]$$



Vector operations

- Vector length/magnitude:

$$|\mathbf{a}| = \sqrt{\mathrm{a_1}^2 + \mathrm{a_2}^2 + \cdots + \mathrm{a_n}^2}$$

- Vector normalization:

$$\hat{\mathbf{a}} = [\frac{\mathrm{a_1}}{|\mathbf{a}|} \quad \frac{\mathrm{a_2}}{|\mathbf{a}|} \quad \dots \quad \frac{\mathrm{a_n}}{|\mathbf{a}|}]$$

Vector operations

- Dot/scalar product:

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a} = \mathrm{a}_1\mathrm{b}_1 + \mathrm{a}_2\mathrm{b}_2 + \ \dots + \mathrm{a}_n\mathrm{b}_n$$

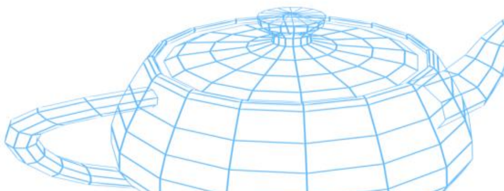
$$\mathbf{a} \cdot \mathbf{0} = 0 \qquad\qquad \mathbf{0} \cdot \mathbf{0} = 0$$

  - If **a** and **b** are **normalized**, then $\cos^{-1}(\mathbf{a} \cdot \mathbf{b})$ is the angle between the two vectors.

- Cross/vector product (for 3D vectors):

$$\mathbf{a} \times \mathbf{b} = [\mathrm{a}_2\mathrm{b}_3 - \mathrm{a}_3\mathrm{b}_2 \quad \mathrm{a}_3\mathrm{b}_1 - \mathrm{a}_1\mathrm{b}_3 \quad \mathrm{a}_1\mathrm{b}_2 - \mathrm{a}_2\mathrm{b}_1]$$

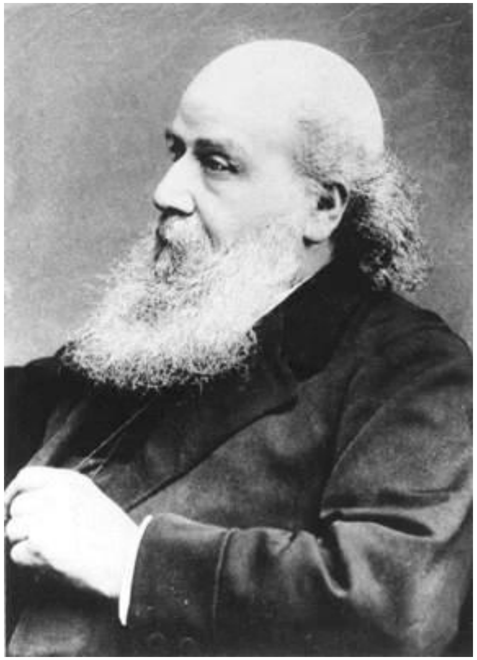
$$\mathbf{a} \times \mathbf{b} \neq \mathbf{b} \times \mathbf{a} \qquad\qquad \mathbf{a} \times \mathbf{a} = \mathbf{0}$$



Matrix notation (1850)

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

James Joseph Sylvester
1814 - 1897

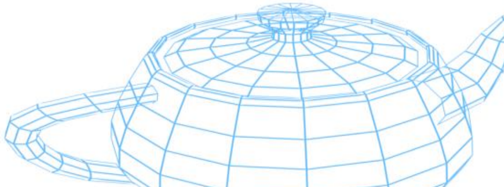


Matrix

$$\begin{bmatrix} 1 & 3.14 & 0 \\ -3.14 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Points and vectors:
  → column/row matrices.

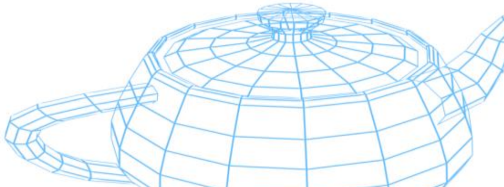
- Operations on points and vectors:
  → rectangular matrices.



Matrix

$$\text{rows}\quad \begin{bmatrix} 1 & 3.14 & 0 \\ -3.14 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{array}{c} \text{columns} \\ \begin{bmatrix} 1 & 3.14 & 0 \\ -3.14 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$



Matrix

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

- Referred to as *nrOfRows* x *nrOfColumns* matrices:
  - e.g., 3x3, 4x4, 2x2, 2x4, 3x2, 4x3, …
- Named using a bold uppercase letter (**A**, **M**, **R**, …):
  - For clarity also $\mathbf{A}_{3x3}$, $\mathbf{M}_{4x4}$, $\mathbf{R}_{4x4}$

Matrix

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

- A matrix with the same number of rows and columns is called **square matrix** (e.g., 2x2, 3x3, 4x4, …).
- The main diagonal is defined by the elements e_{ij} with i=j

Matrix operations

- Matrix by vector (column matrix) multiplication (post-multiplication):

$$\mathbf{A}_{mxn}\mathbf{b}_{nx1} = \mathbf{r}_{mx1}$$

  - defined only if $\mathbf{A}_{mxn}$ and $\mathbf{b}_{nx1}$, i.e., if the number of columns in matrix **A** is equal to the number of rows in vector **b**

Matrix operations

- Vector (row matrix) by matrix multiplication (pre-multiplication):

$$\mathbf{b}_{1\mathrm{x}m}\mathbf{A}_{m\mathrm{x}n} = \mathbf{r}_{1\mathrm{x}n}$$

Matrix operations

- Row matrix by matrix:
  - vector to the left:

$$\mathbf{b}_{1xm}\mathbf{A}_{mxn} = \mathbf{r}_{1xn}$$

- Column matrix by matrix:
  - vector to the right:

$$\mathbf{A}_{nxm}\mathbf{b}_{mx1} = \mathbf{r}_{nx1}$$

Matrix operations

$$\mathbf{A}_{3x3}\mathbf{b}_{3x1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 \\ a_{21}b_1 + a_{22}b_2 + a_{23}b_3 \\ a_{31}b_1 + a_{32}b_2 + a_{33}b_3 \end{bmatrix}$$

$$\mathbf{b}_{1x3}\mathbf{A}_{3x3} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} =$$

$$\begin{bmatrix} b_1a_{11} + b_2a_{21} + b_3a_{31} & b_1a_{12} + b_2a_{22} + b_3a_{32} & b_1a_{13} + b_2a_{23} + b_3a_{33} \end{bmatrix}$$

Matrix operations

- Matrix by matrix multiplication:
  - same rules as before.
  - row and column vectors are a special case of matrix.
  - warning (in general):

$$\mathbf{AB} \neq \mathbf{BA}$$

Matrix operations

$$\mathbf{A}_{3x3}\mathbf{B}_{3x3} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{bmatrix}$$

Matrix operations

- Matrix transpose:
  - turns columns into rows and rows into columns.
  - noted as $\mathbf{A}^{\mathrm{T}}$

$$\begin{bmatrix} 1 & 2 & 0.5 \\ -1 & 3 & -2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & -1 \\ 2 & 3 \\ 0.5 & -2 \end{bmatrix} \qquad \begin{bmatrix} a & b & c \\ e & f & g \\ h & i & j \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} a & e & h \\ b & f & i \\ c & g & j \end{bmatrix}$$

$$(\mathbf{A}^{\mathrm{T}})^{\mathrm{T}} = \mathbf{A}$$

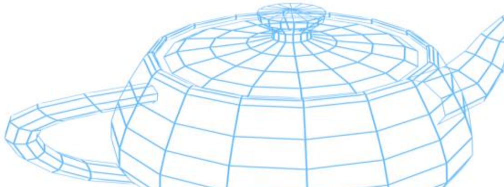
Zero matrix

- Defined by **0** (bold zero):

$$\mathbf{0}= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- Properties:

$$\mathbf{0A} = \mathbf{0}$$



Identity matrix

- All zeros but ones on the main diagonal:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Properties:

$$\mathbf{AI} = \mathbf{IA} = \mathbf{A} \qquad \mathbf{I}^{\mathrm{T}} = \mathbf{I}$$

Matrix operations

- Matrix inverse:
  - the **inverse** of a matrix **A** is noted $\mathbf{A}^{-1}$ and is such that:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

  - not all the matrices have an inverse matrix, e.g.:

$$\mathbf{0}\mathbf{0}^{-1} \neq \mathbf{I}$$

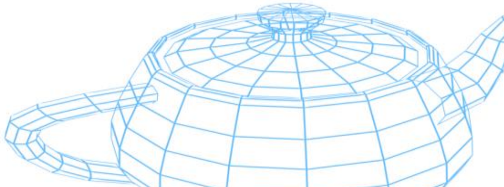
  - a matrix without inverse matrix is called **singular**.

Matrix operations

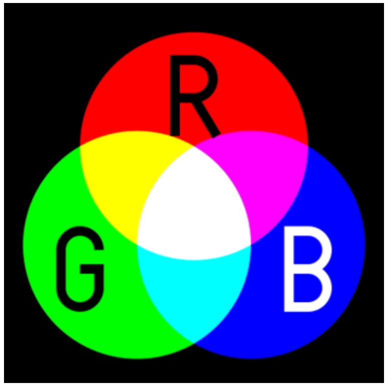
- Matrix inverse:
  - complex operation on large matrices.
  - first compute the determinant:
    - if the determinant is equal to 0, the matrix is singular.

- E.g., Cayley-Hamilton method:

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\left[\frac{1}{6}\left((\mathrm{tr}\mathbf{A})^3 - 3\mathrm{tr}\mathbf{A}\mathrm{tr}\mathbf{A}^2 + 2\mathrm{tr}\mathbf{A}^3\right)\mathbf{I} - \frac{1}{2}\mathbf{A}\left((\mathrm{tr}\mathbf{A})^2 - \mathrm{tr}\mathbf{A}^2\right) + \mathbf{A}^2\mathrm{tr}\mathbf{A} - \mathbf{A}^3\right]$$



```
Matrix inverse() // The OpenGL-way
{
   Matrix t;
   const float fDetInverse = 1.0f / ((_11 * (_22 * _33 - _23 * _32)) -
                                     (_12 * (_21 * _33 - _23 * _31)) +
                                     (_13 * (_21 * _32 - _22 * _31)));
   t._11 =  fDetInverse * (_22 * _33 - _23 * _32);
   t._12 = -fDetInverse * (_12 * _33 - _13 * _32);
   t._13 =  fDetInverse * (_12 * _23 - _13 * _22);
   t._14 =  0.0f;
   t._21 =  -fDetInverse * (_21 * _33 - _23 * _31);
   t._22 =   fDetInverse * (_11 * _33 - _13 * _31);
   t._23 =  -fDetInverse * (_11 * _23 - _13 * _21);
   t._24 =  0.0f;
   t._31 =  fDetInverse * (_21 * _32 - _22 * _31);
   t._32 = -fDetInverse * (_11 * _32 - _12 * _31);
   t._33 =  fDetInverse * (_11 * _22 - _12 * _21);
   t._34 =  0.0f;
   t._41 = -(_41 * t._11 + _42 * t._21 + _43 * t._31);
   t._42 = -(_41 * t._12 + _42 * t._22 + _43 * t._32);
   t._43 = -(_41 * t._13 + _42 * t._23 + _43 * t._33);
   t._44 =  1.0f;
   return t;
}
```



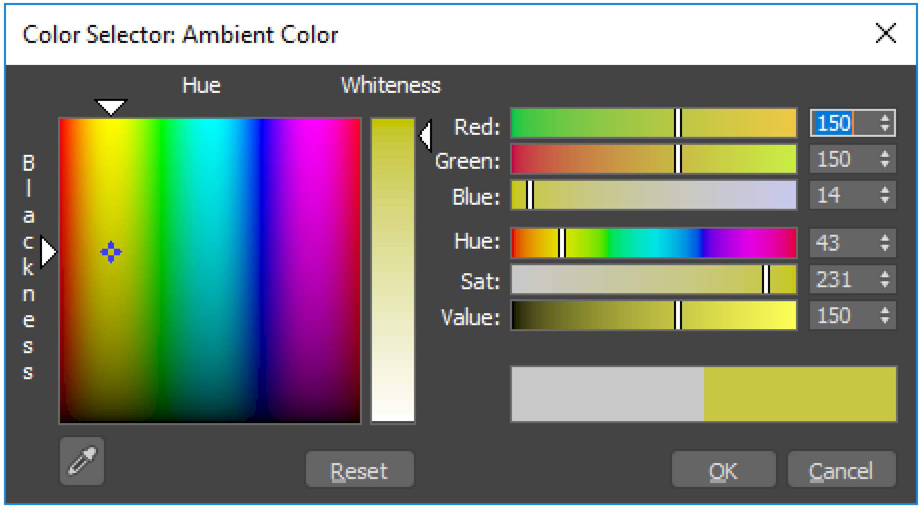
RGB

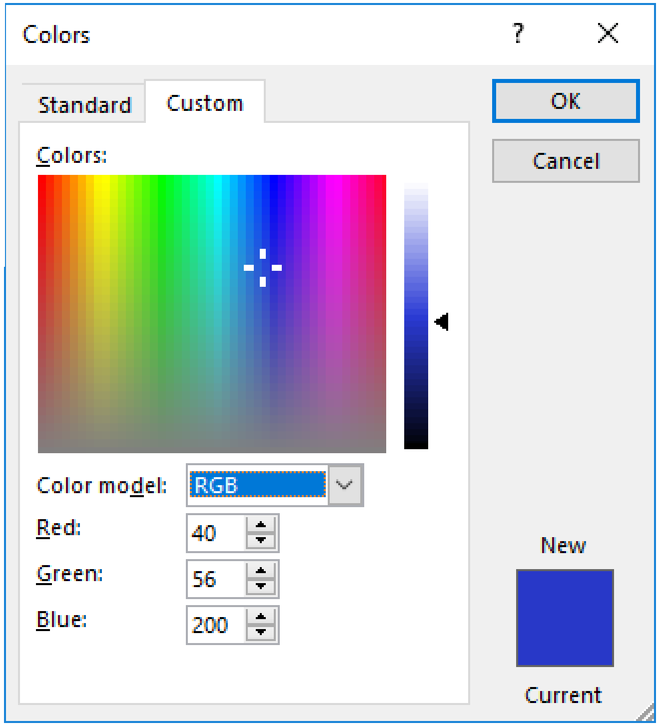
- **R**ed **G**reen **B**lue :
  - Color model used to express colors based on the intensity of the three base colors.
  - Works for light-emitting sources (and not for painting).
  - Additive method:

    $$color = R_{intensity} + G_{intensity} + B_{intensity}$$

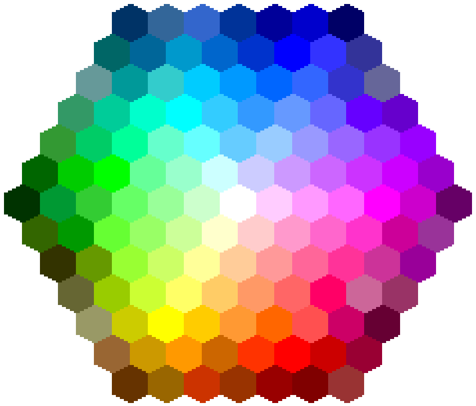
  - Several ways to encode values:
    - Bytes [0-255], e.g.: [255 0 0], [0 255 0], [128 128 128]
    - Float [0.0-1.0], e.g.: [1.0 0.0 0.0], [0.0 1.0 0.0], [0.5 0.5 0.5]
    - Hexadecimal [00-FF], e.g.: #FF0000, #00FF00, #7F7F7F

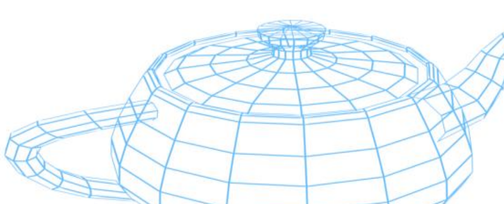

Choose Color
R: 255
G: 0
B: 0
RGB
FF0000
Choose
Cancel


Color Selector: Ambient Color
Hue
Whiteness
Blackness
Red: 150
Green: 150
Blue: 14
Hue: 43
Sat: 231
Value: 150
Reset
OK
Cancel


Colors
Standard
Custom
OK
Cancel
Colors:
Color model: RGB
Red: 40
Green: 56
Blue: 200
New
Current


#f2d91c



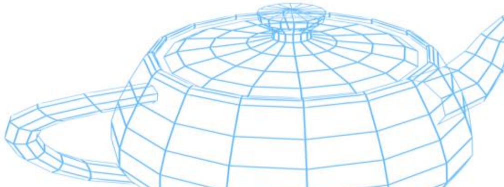


RGBA

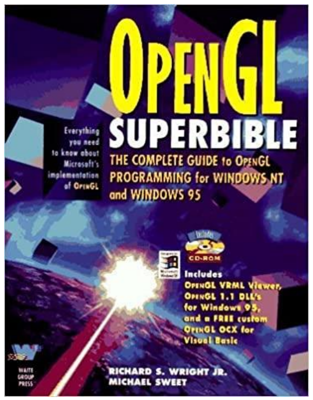
- Another channel (alpha) is added for storing additional information (**usually** transparency):
  - As already seen for the intensities of the other channels, the alpha channel intensity is defined as *0 = transparent*, and *max value = completely solid.*

- RGBA using float = 4 * sizeof(float) = 16 bytes = 128 bit:
  - Good for memory alignment.
  - Perfect for SIMD 128 bit registers.

- Specifying transparent alpha values **will not** automatically activate transparency in OpenGL!



Bibliography



Wright, R. S.
Sweet, M.
OpenGL Superbible
Waite Group

Chapter 2: 3D Graphics Fundamentals

Chapter 7: Manipulating 3D Space: Coordinate Transformations

Tutorials

Central Connecticut University, tutorial on vector math for 3D Computer Graphics (including examples and exercises):

http://chortle.ccsu.edu/VectorLessons/index.html

GLM

- Open**GL M**athematics (GLM) is a C++ math library specifically written with OpenGL in mind:
  - it adopts the same conventions and standards.
  - supports several OSs and compilers.

- It already implements all the necessary functions required by OpenGL (and more):
  - vector classes of various dimensions and types.
  - matrix classes of various dimensions and types.
  - a series of additional functions:
    - quaternions, math functions and constants, deprecated OpenGL functions, etc.

GLM

- Current version: **0.9.9.8**

- Available at:
  - http://glm.g-truc.net *(main page)*
  - https://github.com/g-truc/glm/releases *(library code)*
  - https://github.com/g-truc/glm/blob/0.9.9.8/doc/manual.pdf *(doc)*

- Header-only:
  - no **.lib**, **.a**, **.dll**, or **.so** required.
  - just **#include <glm/glm.hpp>**

GLM

- A simple example:

```
#include <glm/glm.hpp>

int foo()
{
  glm::vec3 a(0.0f, 1.0f, -1.0f);
  glm::vec3 b = glm::vec3(1.0f);
  glm::vec3 c = a + b;
  glm::vec3 d = glm::cross(a, c);
  glm::vec4 r = glm::vec4(d, 1.0f);

  glm::mat3 M = glm::mat3(1.0f); // Identity matrix
  M[2] = c;                      // Set third column to ‘c’
  glm::mat3 Z(0);                // Zero matrix
  M = M * Z;

  return 0;
}
```

GLM

- Another simple example (for printing values):

```
#include <glm/glm.hpp>
#include <glm/gtx/string_cast.hpp>

int foo2()
{
  glm::vec3 a(0.0f, 1.0f, -1.0f);
  std::cout << glm::to_string(a) << std::endl;

  glm::mat3 M = glm::mat3(1.0f);
  std::cout << glm::to_string(M) << std::endl;

  return 0;
}
```

GLM

- OpenGL (thus GLM) accesses matrices in column-major order, e.g.:

$$\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

← *in the documentation*

- But C arrays are stored in row-major order:

```
glm::mat3 mat( a, b, c,
               d, e, f,
               g, h, i);
```

← *in the code*

GLM

- No need to create a new class: RGBA = XYZW.
  - Reuse **glm::vec3**, e.g.:

```
// Define red [1.0 0.0 0.0]:
glm::vec3 color;
   color.r = 1.0f;
   color.g = 0.0f;
   color.b = 0.0f;
```

  - …or **glm::vec4** for RGBA colors, e.g.:

```
// Define red with alpha channel [1.0 0.0 0.0 1.0]:
glm::vec4 color;
   color.r = 1.0f;
   color.g = 0.0f;
   color.b = 0.0f;
   color.a = 1.0f;
```