Fundamentals of Computer Graphics and Image Processing

Computer Graphics - Exercise #01

Introduction

Dana Škorvánková

dana.skorvankova@fmph.uniba.sk

Labs - grading, current score...

https://dai.fmph.uniba.sk/w/Course:ZPGSO/en

Outline:

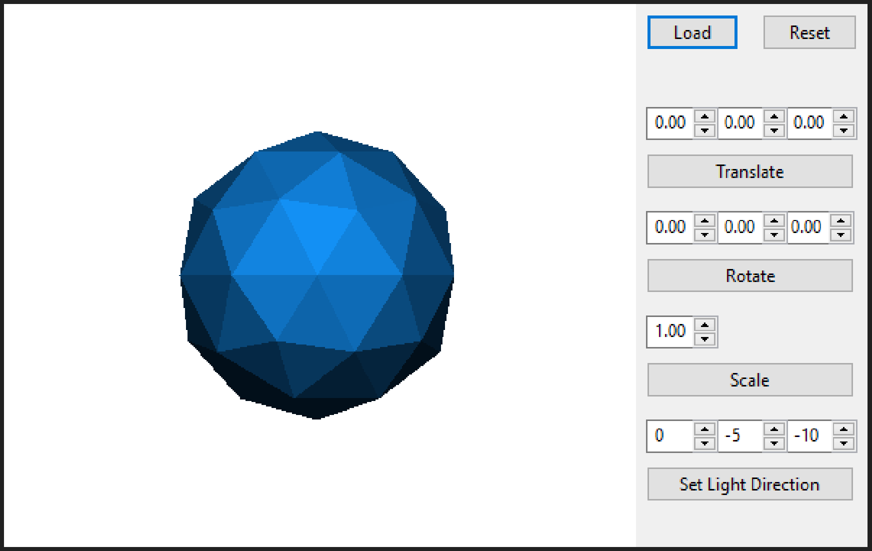
- Project
- 5 exercises + 1 project consultation in the end

Labs Evaluation

Emergency contact (practical questions):
Mgr. Lukáš Gajdošech
l.gajdosech@gmail.com

- A single project split into several stages:

Stage	Description	Evaluation	Deadline
Stage #1	Obj. file loading	3 points	11.10.2021
Stage #2	Transformations	10 points	25.10.2021
Stage #3	Shading and Lighting	7 points	7.11.2021



Computer Graphics

Coco (2017)

The Witcher III (2015)

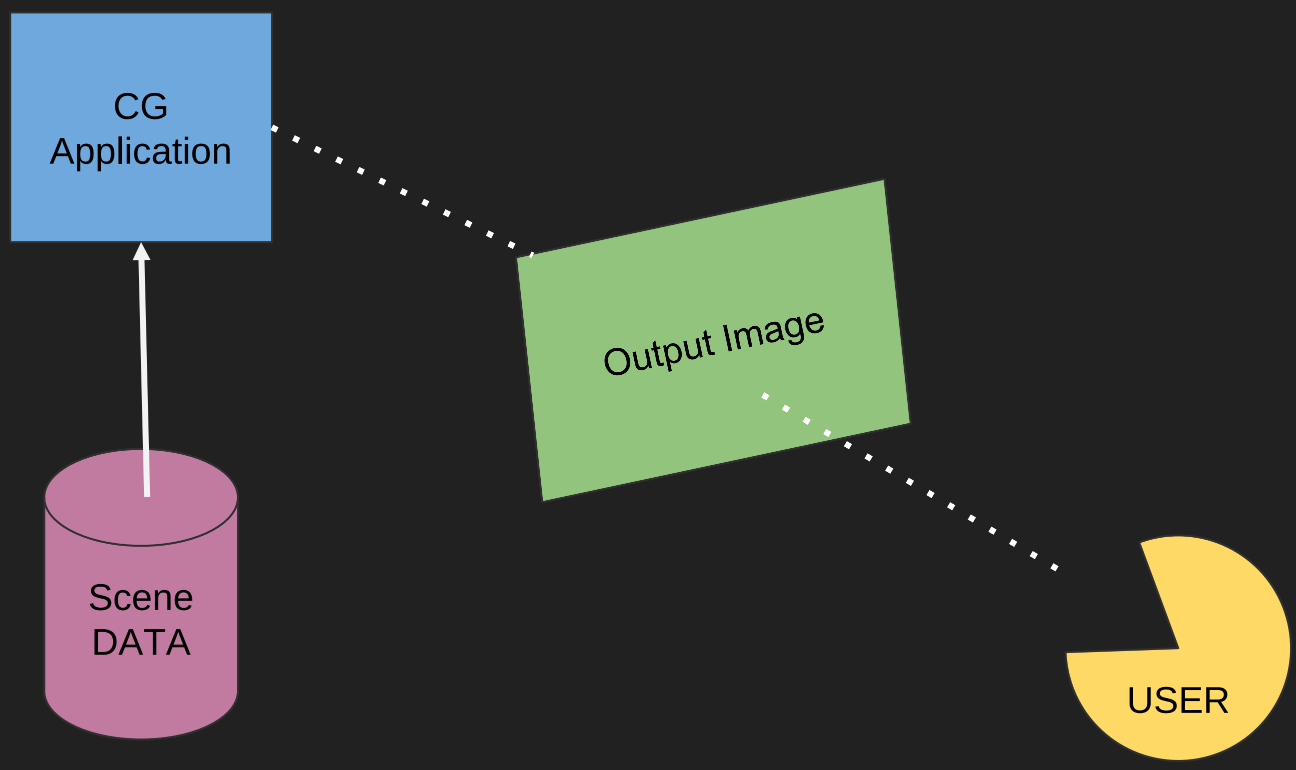
Computer Graphics

What is the difference in CG for games …

… and movies?

Warcraft (2016)

Task of Computer Graphics



Creating imagery of virtual objects/scenes in a way which is consumable by user

Reference Model

Each part may contain unique solutions, connected with standard interfaces

Purpose:

- ➔ Separate modeling and rendering
- ➔ Separate device-dependent and device-independent parts

Application program → **Graphical system** → **Output device**

Time for Linear Algebra

Matrix operations

$$\begin{pmatrix} \boxed{a_{11} \quad a_{12}} \\ a_{21} \quad a_{22} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} \boxed{c_{11}} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$$A = \begin{pmatrix} 3 & 5 \\ 1 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 4 & 1 \\ 2 & -1 \end{pmatrix} \qquad C = \begin{pmatrix} 0 & 2 \\ 2 & 3 \end{pmatrix}$$

I. A * B =

II. B * A =

III. (A * B) * C =

IV. A * (B * C) =

Matrix operations

$$A = \begin{pmatrix} 3 & 5 \\ 1 & 2 \end{pmatrix} \quad B = \begin{pmatrix} 4 & 1 \\ 2 & -1 \end{pmatrix} \quad C = \begin{pmatrix} 0 & 2 \\ 2 & 3 \end{pmatrix}$$

I. $A * B = \begin{pmatrix} 22 & -2 \\ 8 & -1 \end{pmatrix}$ Not Commutative! **II.** $B * A = \begin{pmatrix} 13 & 22 \\ 5 & 8 \end{pmatrix}$

III. $(A * B) * C = \begin{pmatrix} -4 & 38 \\ -2 & 13 \end{pmatrix}$ Associative! **IV.** $A * (B * C) = \begin{pmatrix} -4 & 38 \\ -2 & 13 \end{pmatrix}$

Matrix operations

$$A = \begin{pmatrix} 3 & 5 & 1 \\ 2 & 1 & 0 \\ 1 & 3 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad C = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

I. A * C =

II. C * A =

III. A * B =

Matrix operations

Identity Matrix!

$$A = \begin{pmatrix} 3 & 5 & 1 \\ 2 & 1 & 0 \\ 1 & 3 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad C = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

I. $A * C = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$

II. $\cancel{C * A} =$

III. $A * B = A$

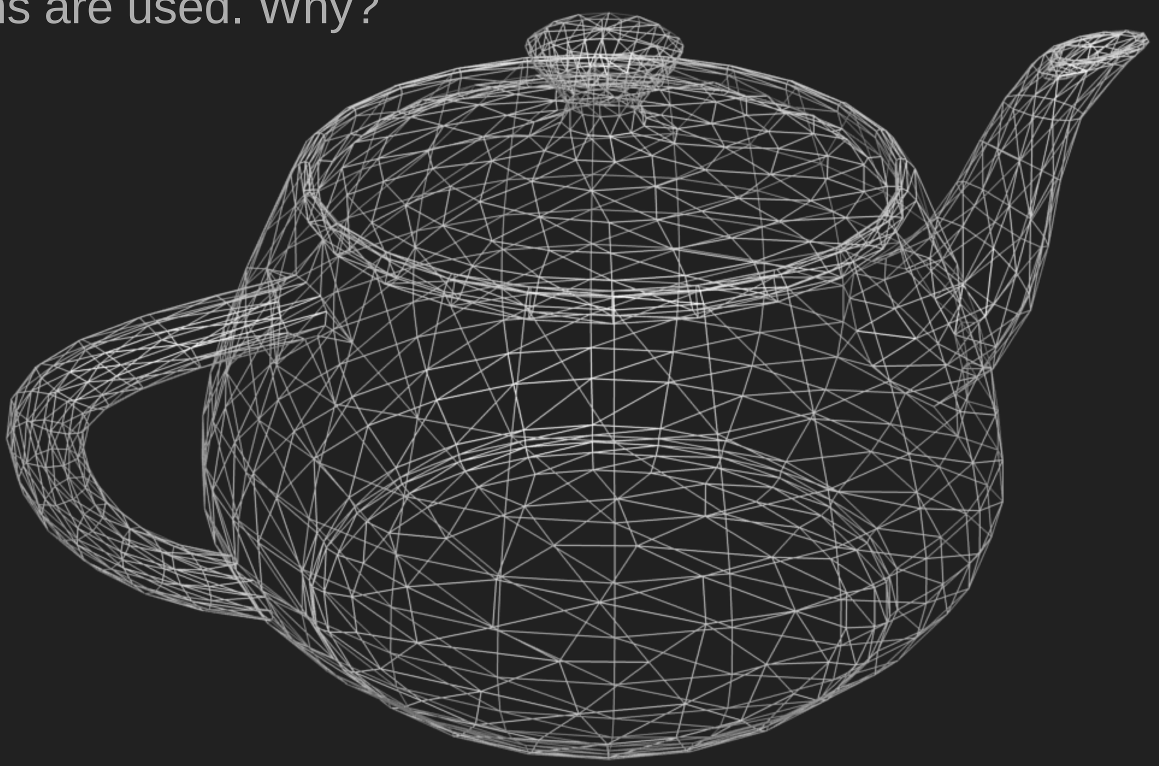
Polyhedral Representation

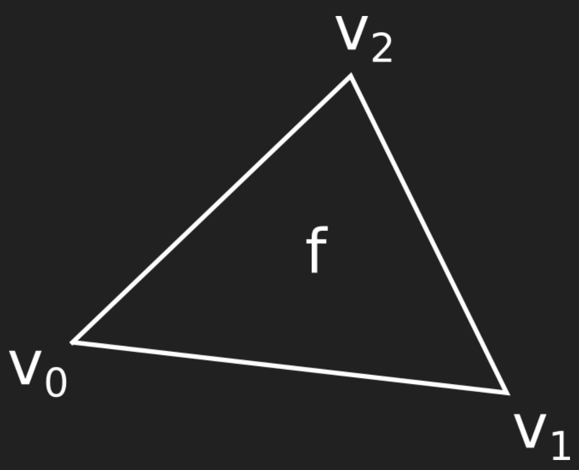
General Polygonal Mesh

Boundary representation of 3D object (polyhedron).

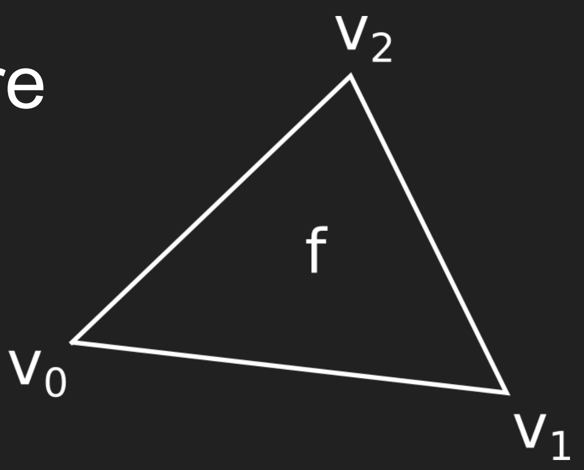
Represented as set of polygons (faces), which are interconnected by vertices and edges.

In CG, only triangular polygons are used. Why?





Triangular “Indexed Face” Structure



Mesh structure contains two lists:

- ➢ An array of vertices - 3D coordinates *x,y,z*
- ➢ An array of integers containing triplets of indices to the array of vertices
    - ○ size of the array is three times the number of faces

Con: *No information of neighboring elements (slow topological algorithms)*

Pro: *Structure best suitable for visualization using graphics cards and parallelization*

Used in file formats: *Collada, 3DS, OBJ, ...*

```
struct Vertex
        Float x, y, z;
        ...

struct Mesh
        Vertex[] vertices;
        Int[] indices;
```

Wavefront (.obj) file structure

```
# Blender v2.67 OBJ File
# www.blender.org
o BuzzLightyear
v -0.7090 1.0177 0.2252
v -0.6857 1.0249 0.0915
v -0.8108 1.0904 0.0497
v -0.4985 0.7124 -0.0398
v -0.5016 0.6992 -0.0638
v -0.5012 0.7177 0.0000
...
f 4 3 71
f 6 3 345
f 348 8 345
f 8 7 345
...
```

Stored in regular text file

First word on each line = a data specifier, telling what kind of data follows:

- *# - comment*
- *o - object/mesh name*
- *v - vertex*
- *f - face*
- *… other specifiers can be found on wikipedia*

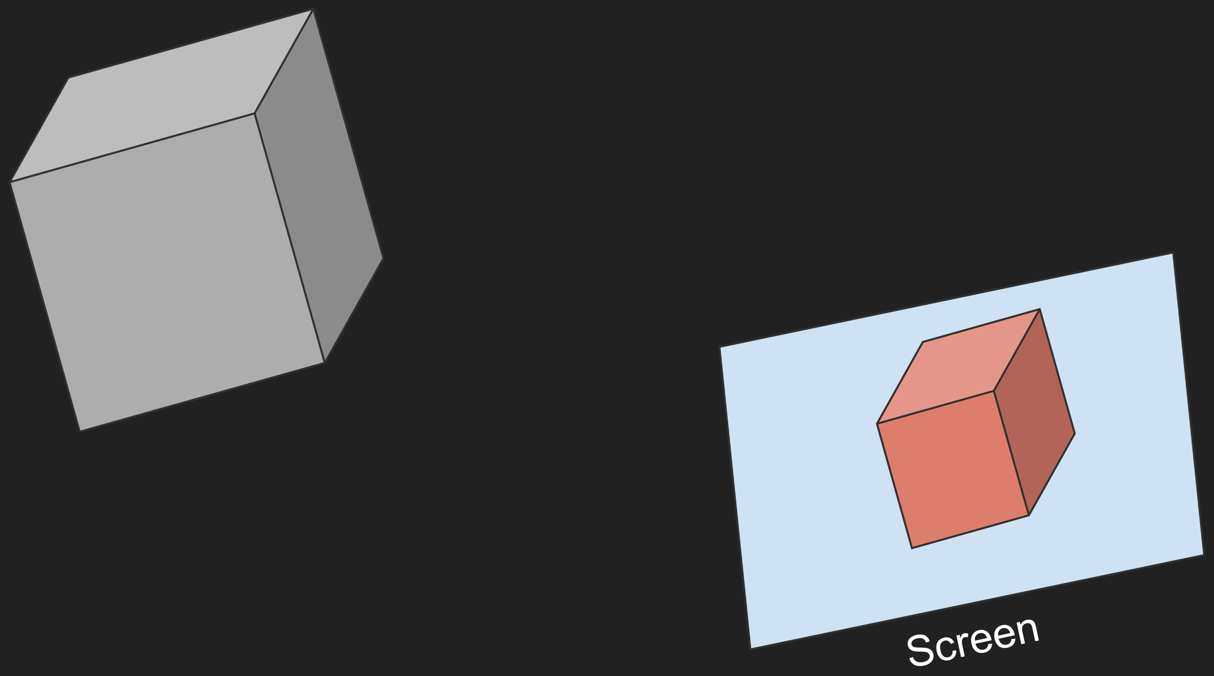
Indexing of vertices starts form 1

We will be using only vertices and TRIANGLE faces in form that is displayed on left

Coordinate Systems

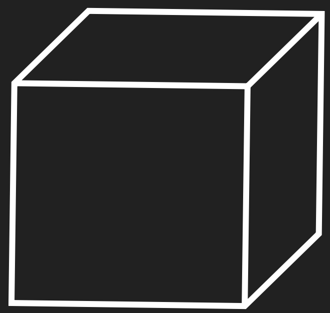
Problem Definition

Having a 3D scene, how to deliver final imagery to user?



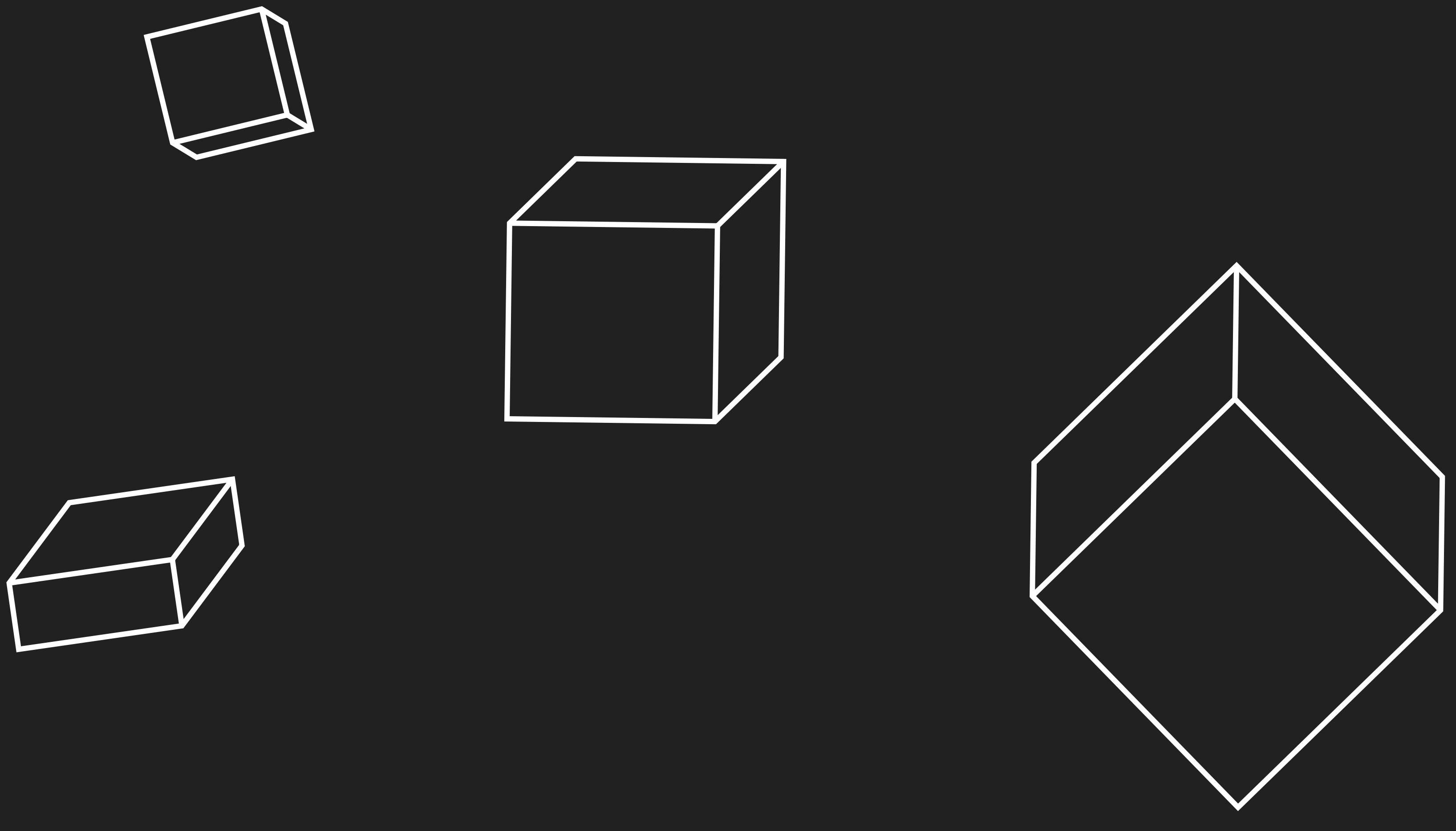
Problem Definition

A polyhedron can be loaded from a file, or procedurally generated.



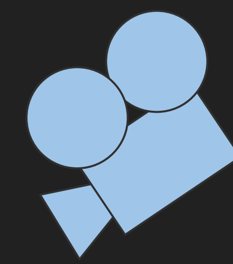
Problem Definition

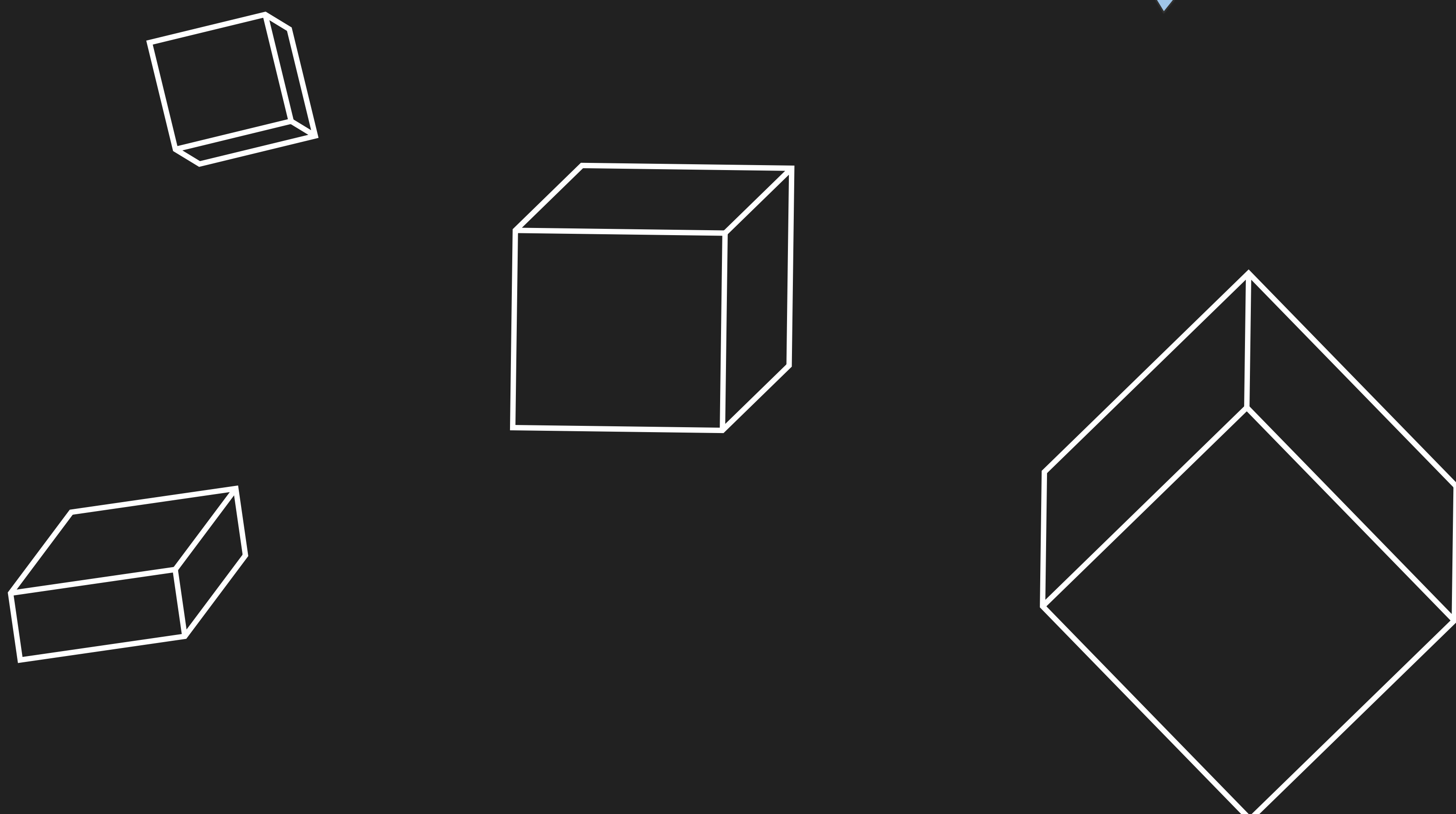
Multiple instances of the same geometry should not be re-defined.



Problem Definition

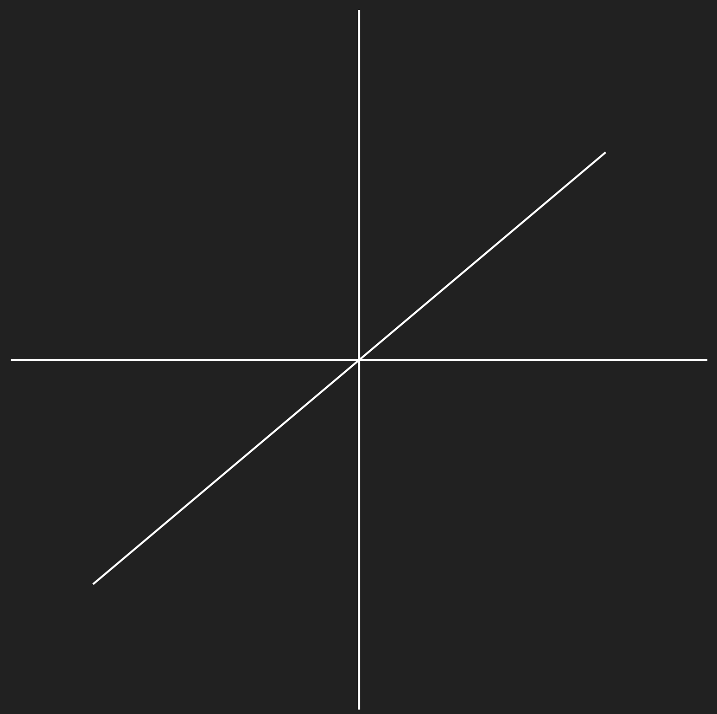
An observer is looking at a scene from a certain view.



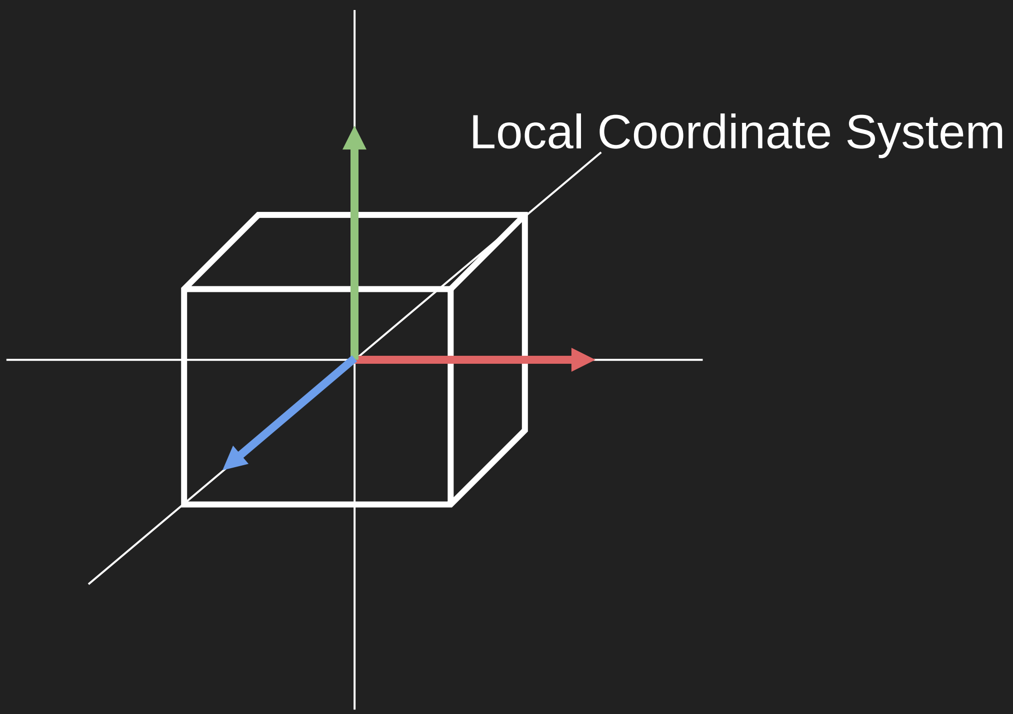


Geometry Space

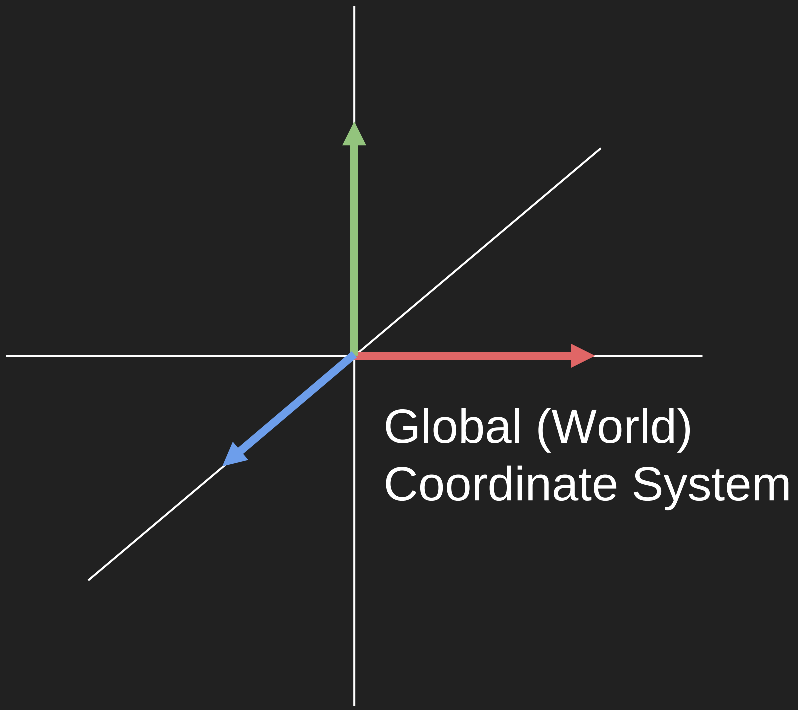
Concept of coordinate systems is used to define relations between objects in scene,



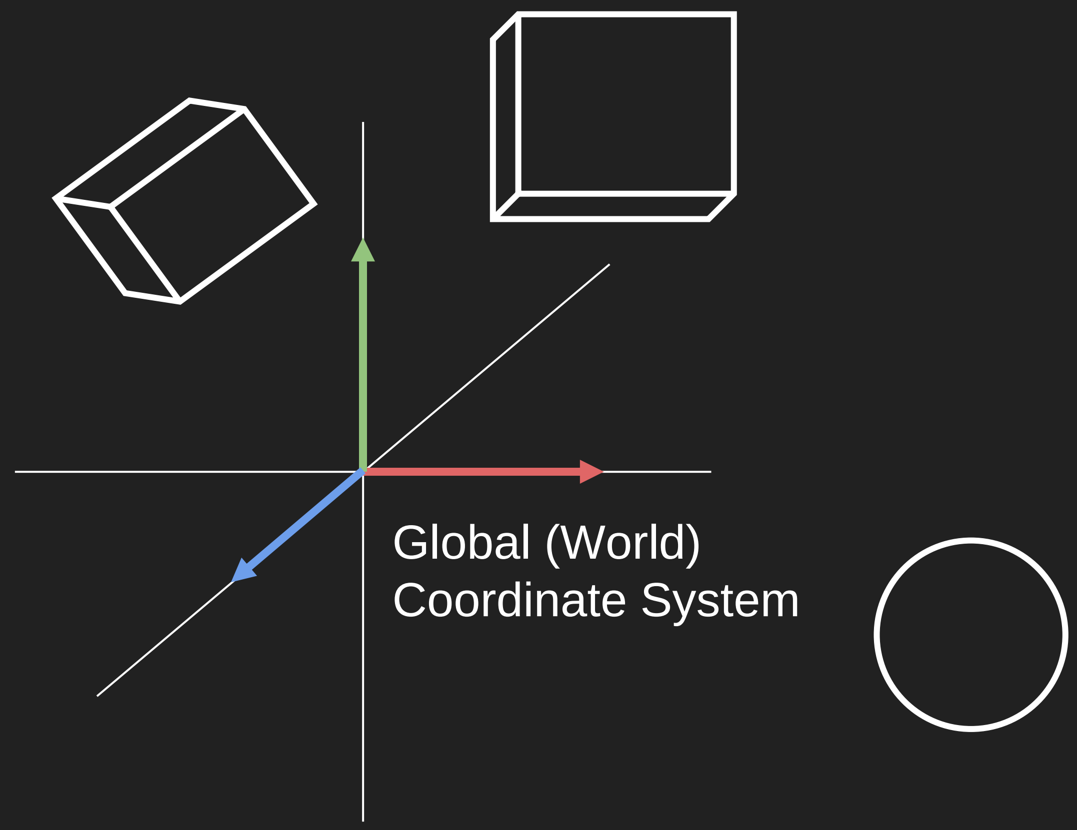
Geometry Space



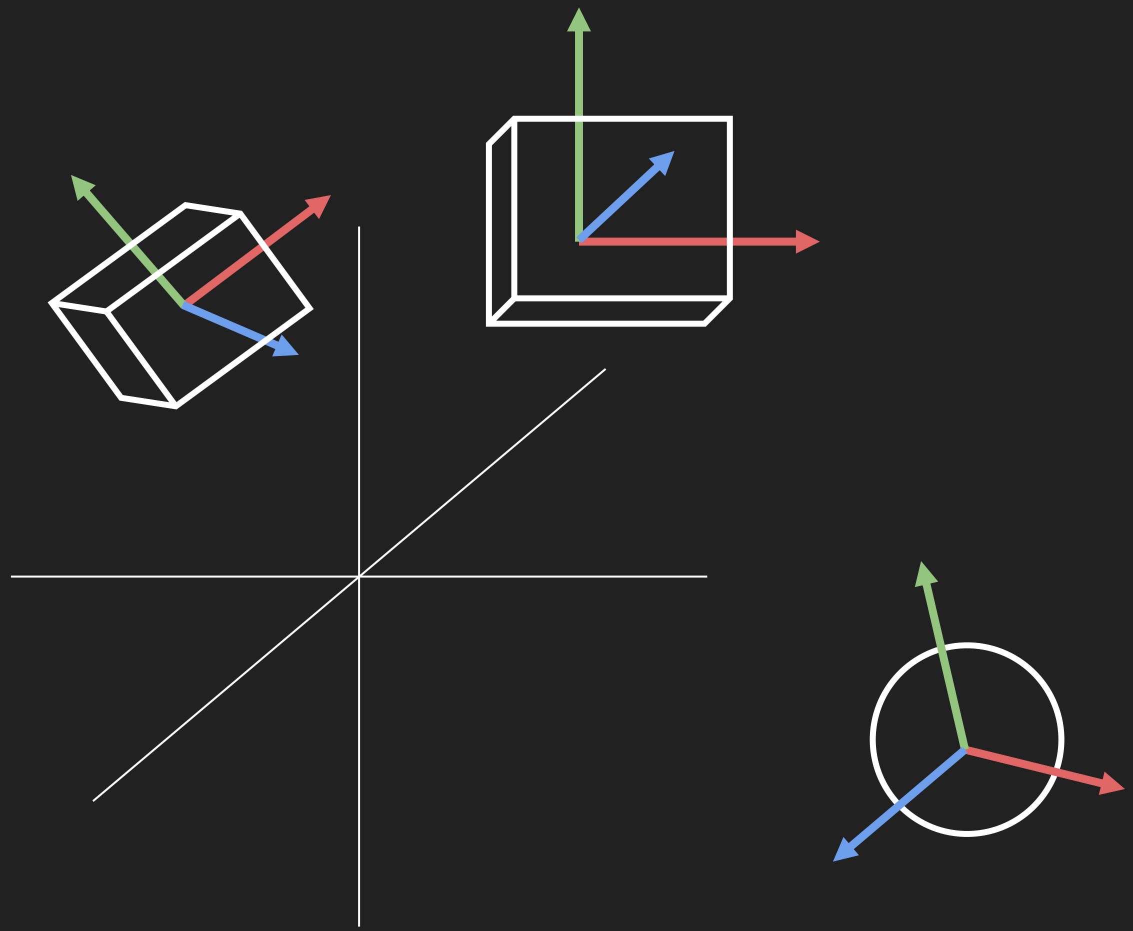
Coordinate Systems



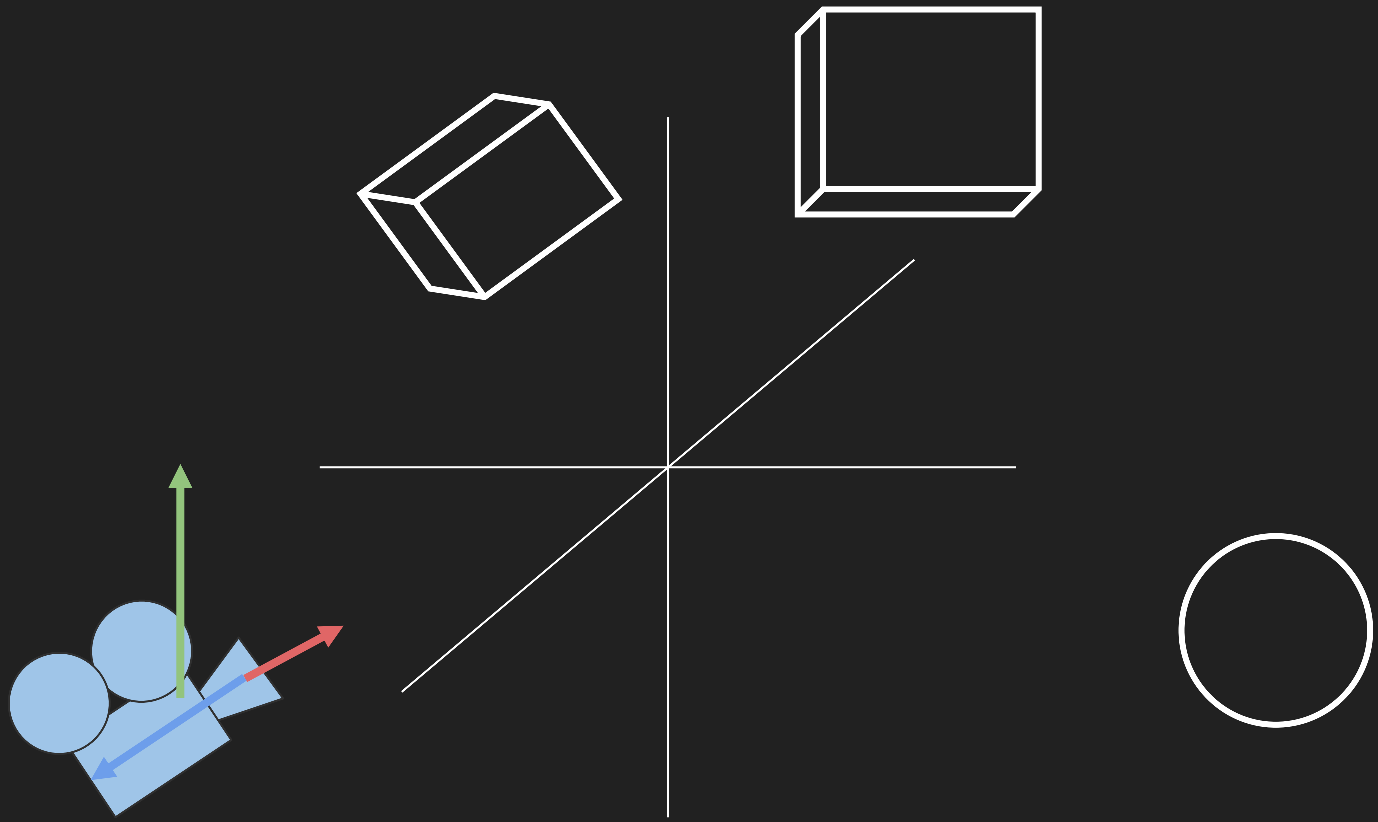
Coordinate Systems



Coordinate Systems



Coordinate Systems

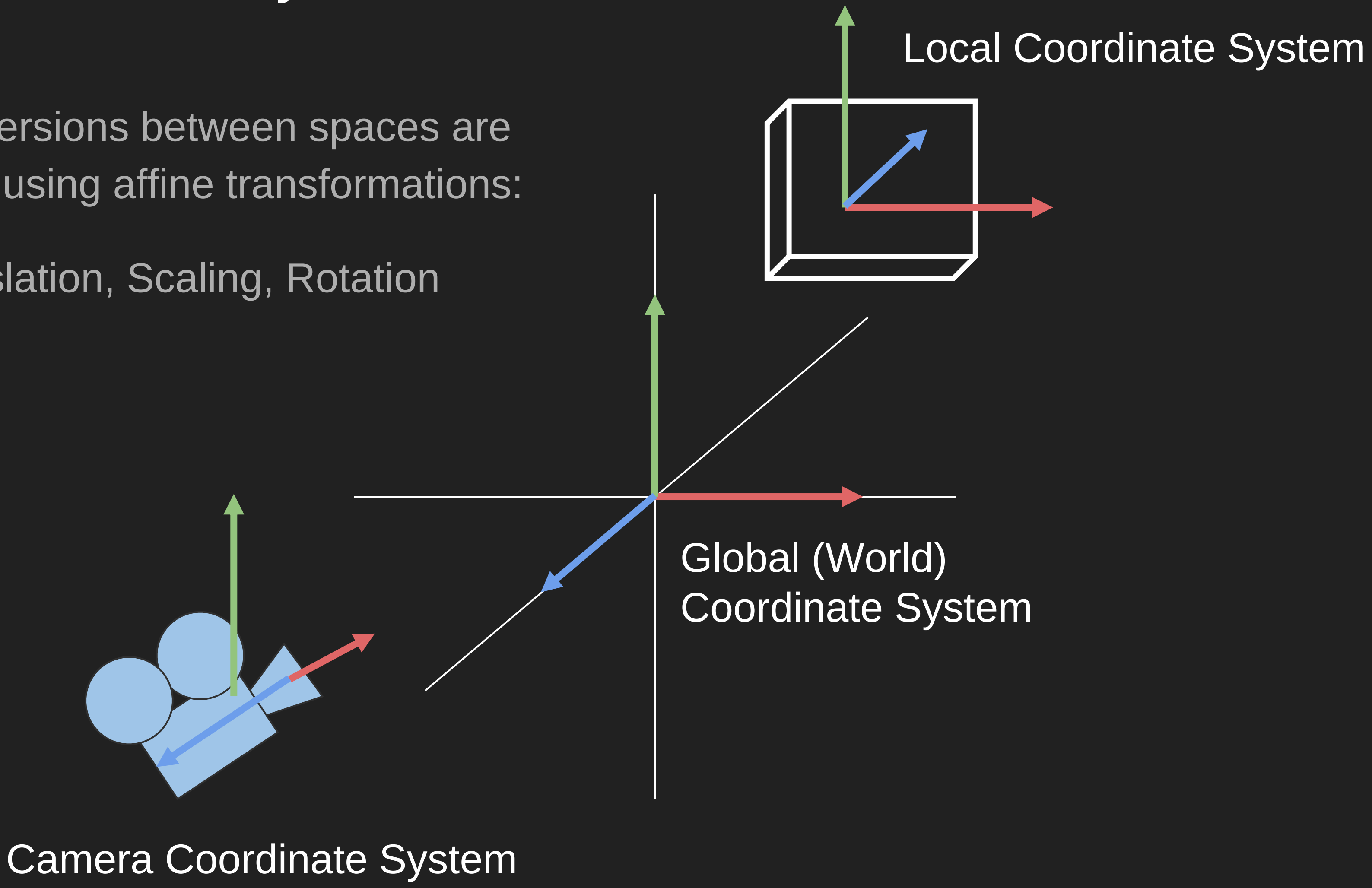


Camera Coordinate System

Coordinate Systems

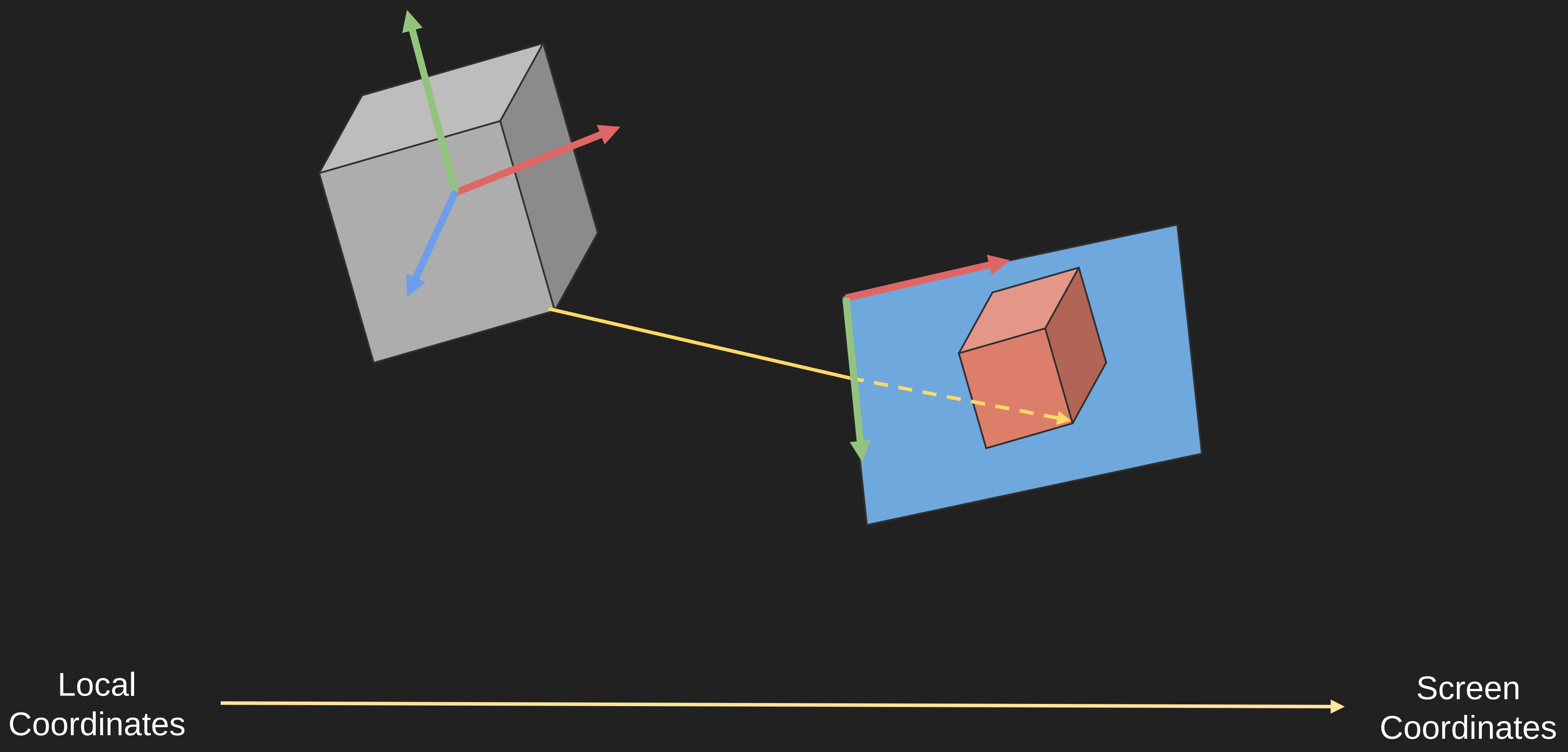
Conversions between spaces are done using affine transformations:

Translation, Scaling, Rotation

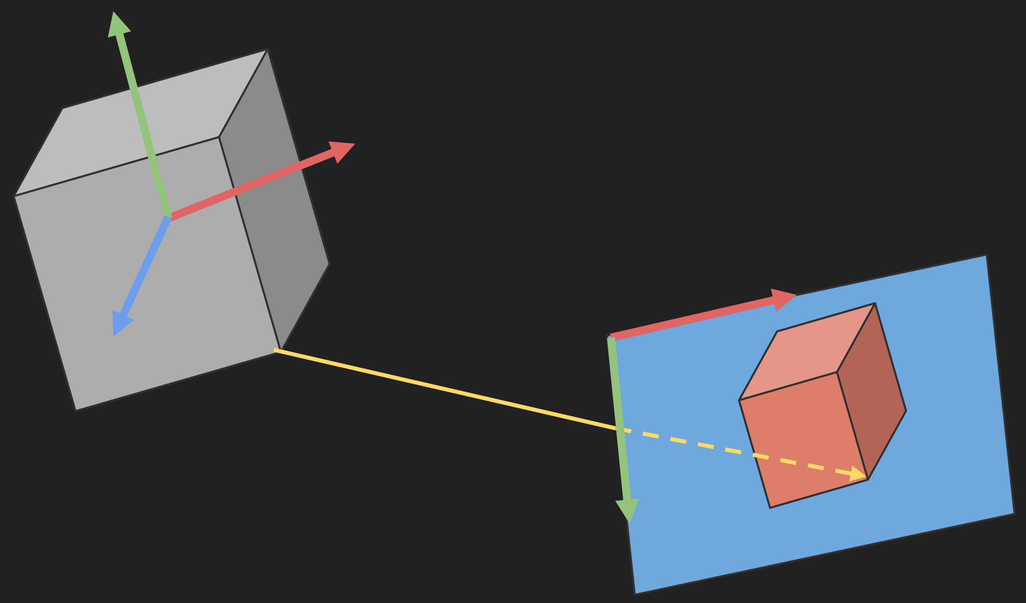


Coordinate Systems

How to get screen position of vertex defined in local coordinates.



Coordinate Systems



Local Coordinates → **Model Matrix** → Global Coordinates → **View Matrix** → Camera Coordinates → **Projection Matrix** → Screen Coordinates

Affine Transformations

Affine Transformation Matrices

A unified and compact way of representing transformations

Multiple transformations (rotation, translation…) can be joined into one matrix

Applying a transformation = Matrix multiplication

GPUs are optimized for matrix operations

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Identity

$$\begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Scaling

Matrix multiplication from left:

Pt = M * P

P	a point
Pt	transformed point P
M	transformation matrix

Affine Transformation Matrices

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Identity

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Scaling

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{Cos}\ \varphi_x & -\mathrm{Sin}\ \varphi_x & 0 \\ 0 & \mathrm{Sin}\ \varphi_x & \mathrm{Cos}\ \varphi_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation X

$$\begin{pmatrix} \mathrm{Cos}\ \varphi_y & 0 & \mathrm{Sin}\ \varphi_y & 0 \\ 0 & 1 & 0 & 0 \\ -\mathrm{Sin}\ \varphi_y & 0 & \mathrm{Cos}\ \varphi_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation Y

$$\begin{pmatrix} \mathrm{Cos}\ \varphi_z & -\mathrm{Sin}\ \varphi_z & 0 & 0 \\ \mathrm{Sin}\ \varphi_z & \mathrm{Cos}\ \varphi_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation Z

Points vs Vectors

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation

Point **(x, y, z, 1)**

Can be translated, scaled and rotated

Vector **(x, y, z, 0)**

Can be scaled and rotated, but is invariant against translation

$$\begin{pmatrix} 1 & 0 & 0 & 52 \\ 0 & 1 & 0 & 18 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}$$

Project Stage #1