CSE-170 Computer Graphics

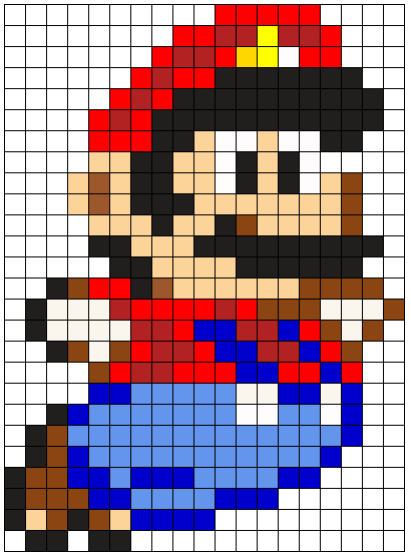
Lecture 2

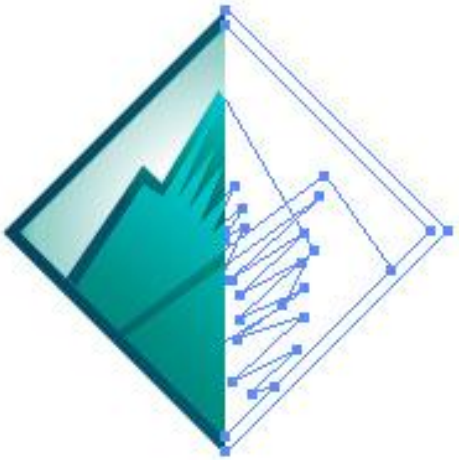
Rendering Pipeline

Dr. Renato Farias
rfarias2@ucmerced.edu

Images

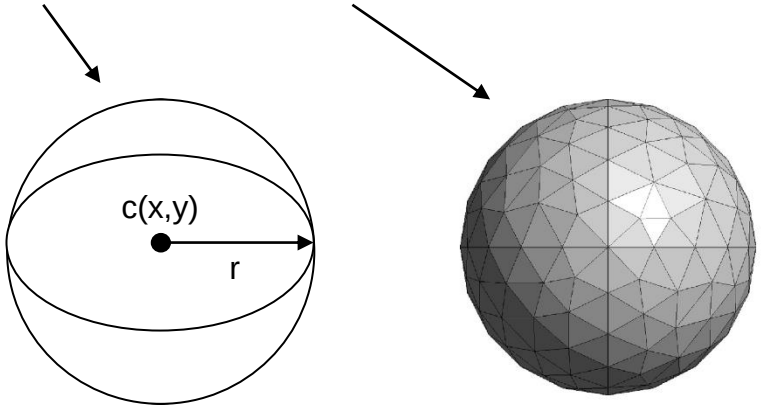
- raster image
  - 2D array of pixel values (red, green, blue color channels)
  - the goal of the rendering pipeline
- vector image
  - description of shapes
  - instructions for displaying image rather than explicitly storing pixel values





3D objects and scenes

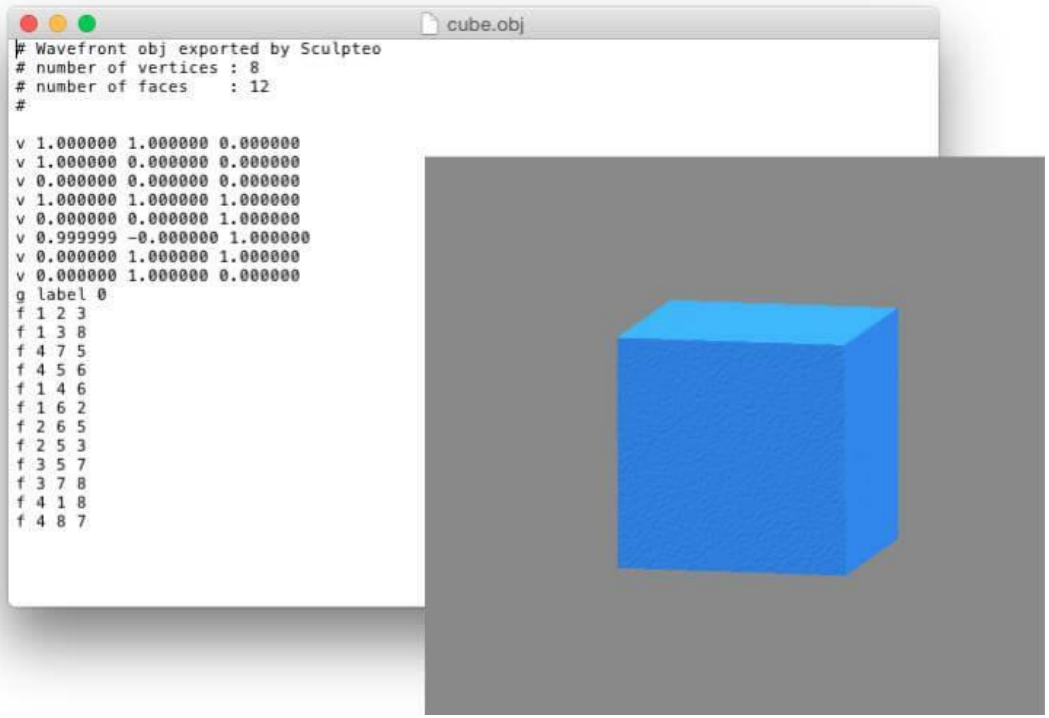
- First step: representation of 3D objects in the computer memory
  - How to represent a sphere?
    - Implicitly or explicitly



  - What about a generic object?
    - Most likely a 3D mesh

3D objects and scenes

- 3D objects are typically stored in files that specify their geometry and all of their attributes (for ex, .obj files)
  - Vertices, how those vertices are connected (faces), normal, texture coordinates...



Approaches to Rendering

How to generate images of 3D scenes?

- Goal: to create an image (rendering)
- Start with objects, end with updating the pixels in the image
- Main key approaches:

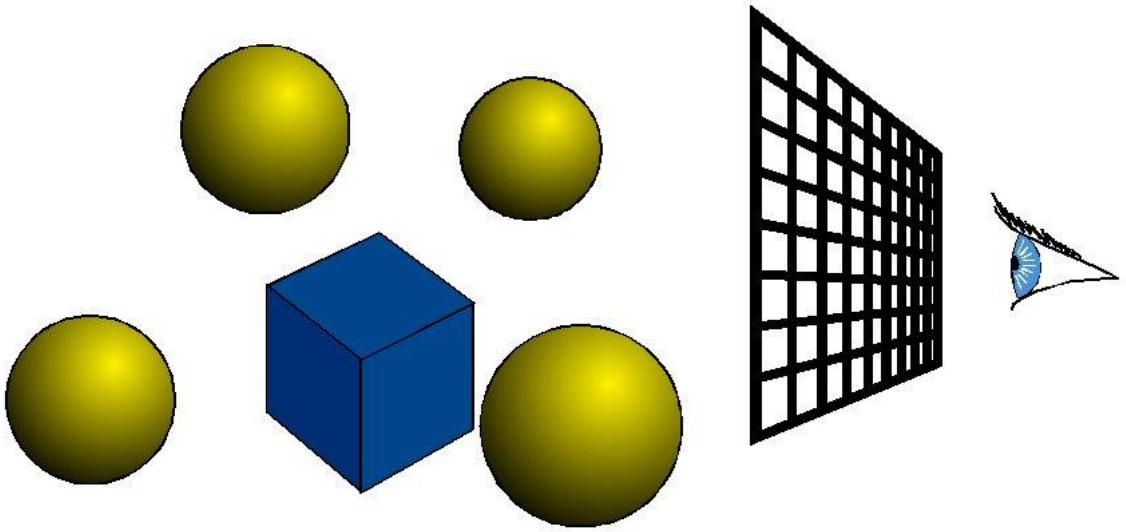
  1) Ray Casting / Ray Tracing (more realistic rendering, but slow)

  2) Rasterization / Object-Ordering Rendering (the rendering pipeline for interactive graphics, efficient and widely used)

(P.S.: there are also other more advanced approaches for realistic rendering, like radiosity and photon mapping methods)

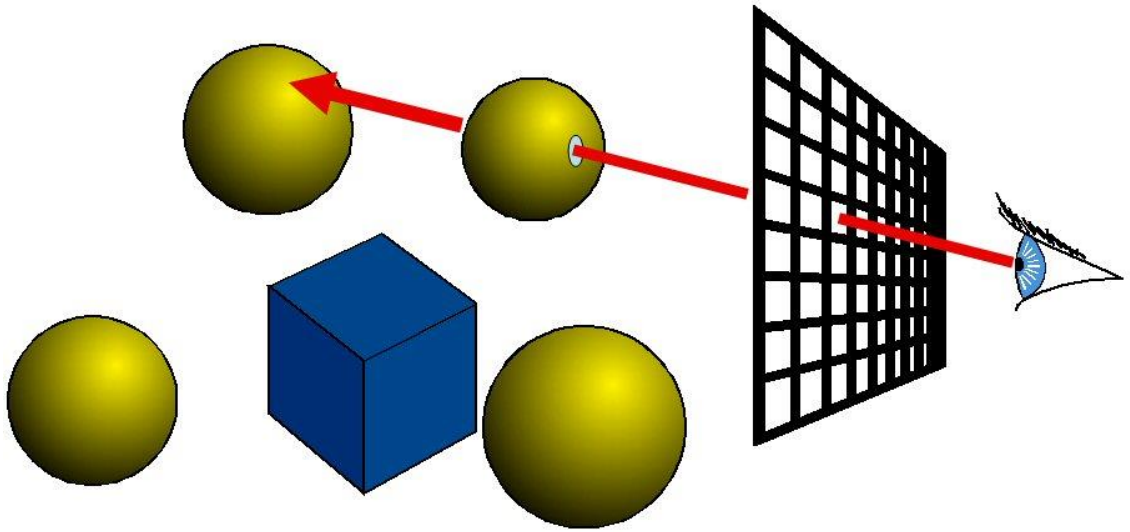
Ray Casting

- for every pixel construct a ray from the eye
  - for every object in the scene
    - find intersection with the ray
    - keep color if closest



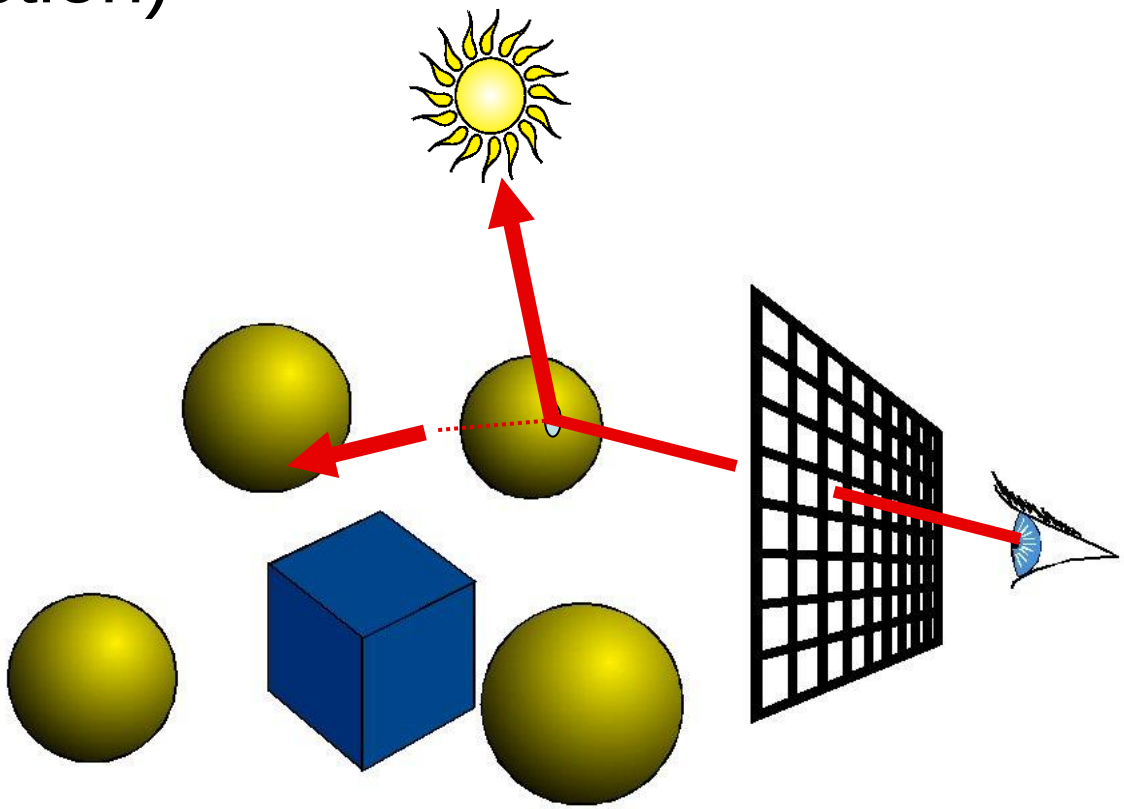
Ray Casting

- for every pixel construct a ray from the eye
  - for every object in the scene
    - find intersection with the ray
    - keep color if closest
      (automatic visible-surface determination)



Ray Tracing

- Shading (interaction of light and material)
- Secondary rays (shadows, reflection, refraction)

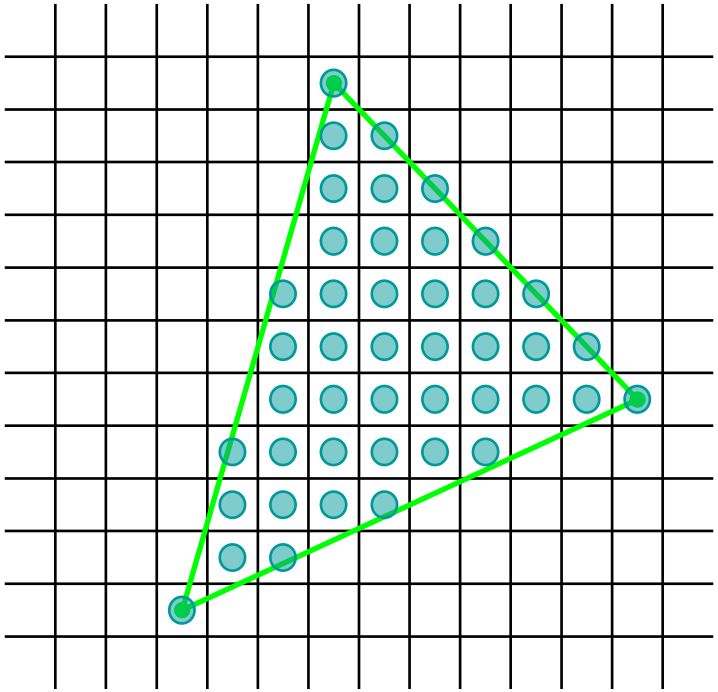


Ray Tracing

- Persistence of Vision Raytracer (povray.org)

Rasterization

- for every object in the scene
  - for each triangle
    - find all pixels in the image the triangle occupies (rasterization)
    - keep color of closest object to viewer



Rendering approaches

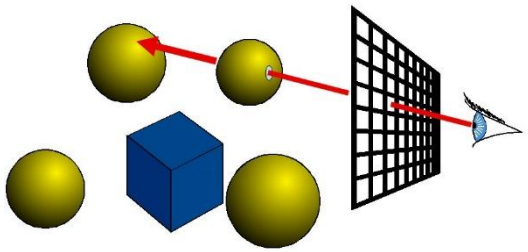
Ray Casting / Tracing

Pros:

- more realistic rendering

Cons:

- must cast a ray for every pixel, even if it results in no intersections
- repeatedly searches scene to find intersection with objects



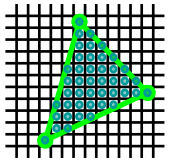
Rasterization

Pros:

- efficient (single pass through scene visiting each object once)
- hardware-accelerated

Cons:

- visible-surface determination requires more consideration
- harder to do effects like shadows, reflections, refractions, etc.



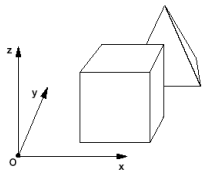
The Rendering Pipeline

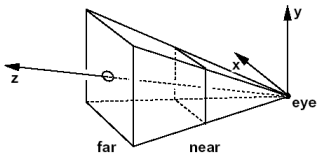
The Rendering Pipeline

- The rendering pipeline is a conceptual model of the steps required to render a 3D scene onto a 2D screen
- Rasterization is the most common approach for real-time applications, and what the rendering pipeline (and GPUs) focus on
- It's not specific to OpenGL (or any other graphics API)

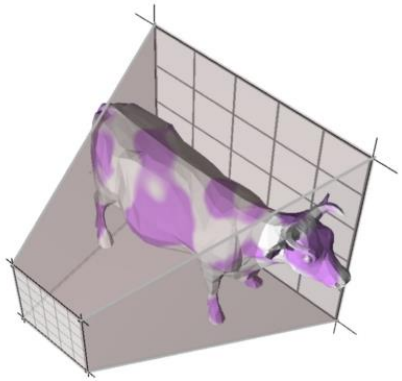
The Rendering Pipeline

- Transformations

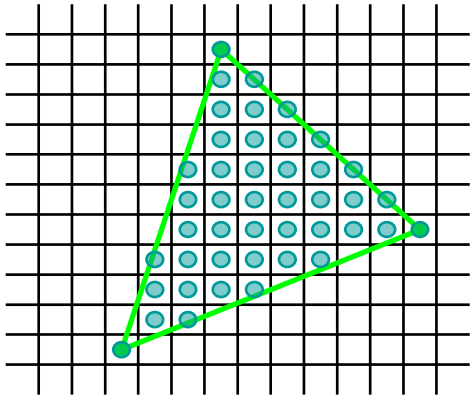




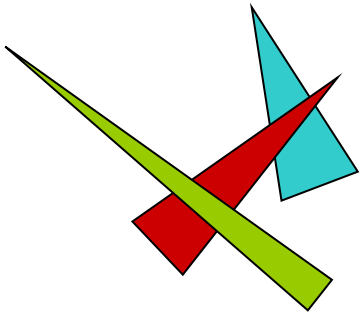
- Clipping



- Rasterization

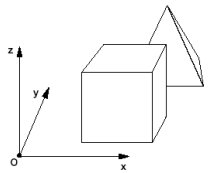


- Visibility



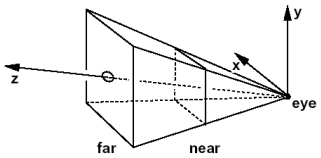
The Rendering Pipeline

Modeling Transformations

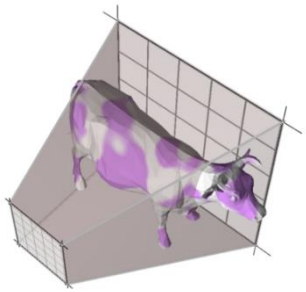


Lighting

Viewing/Camera Transformations

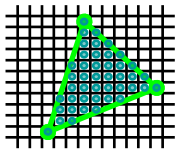


Projection Transformation



Clipping

Scan Conversion (or rasterization)

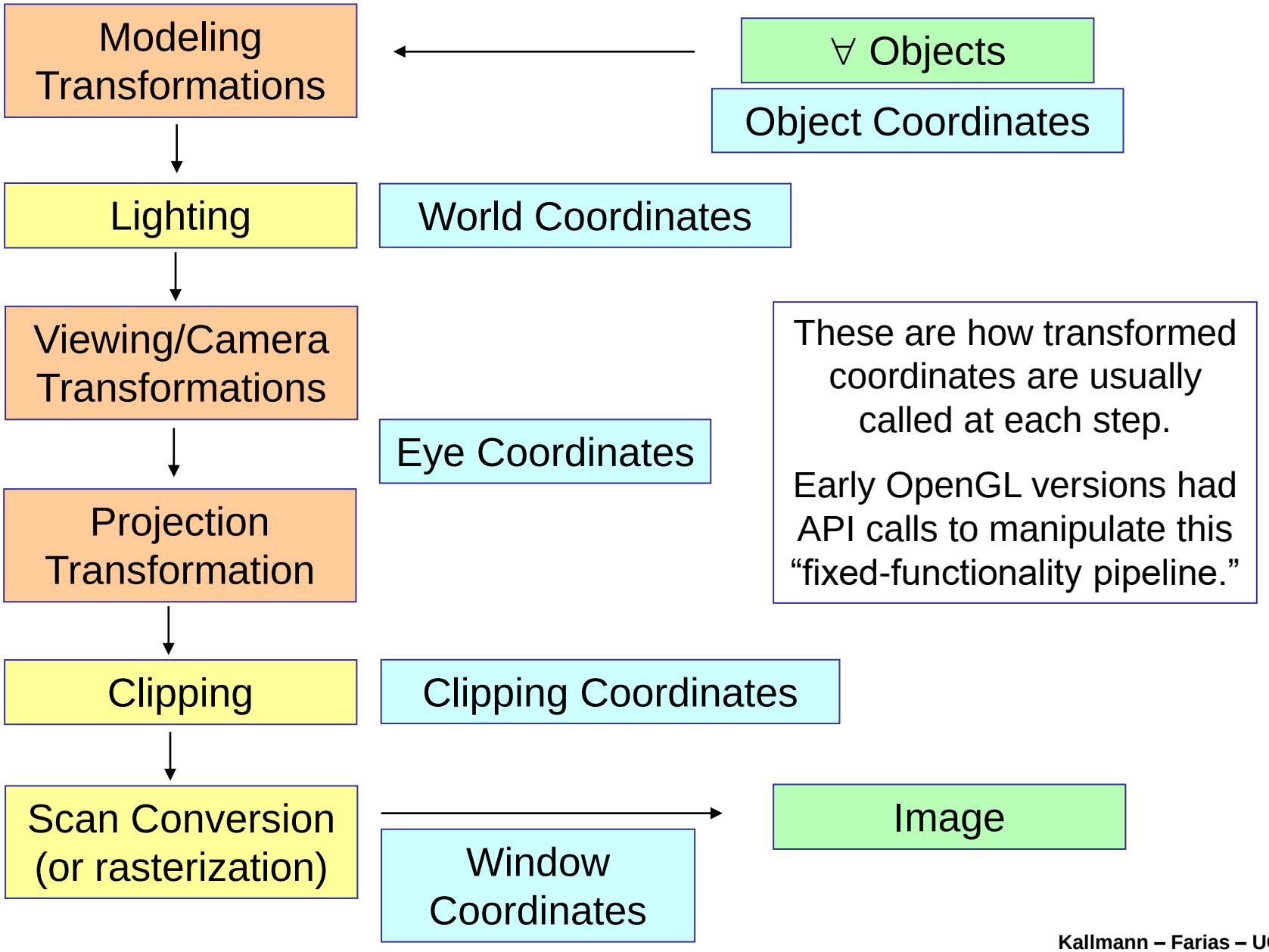


∀ Objects

These are the usual operations in the rendering pipeline

Image

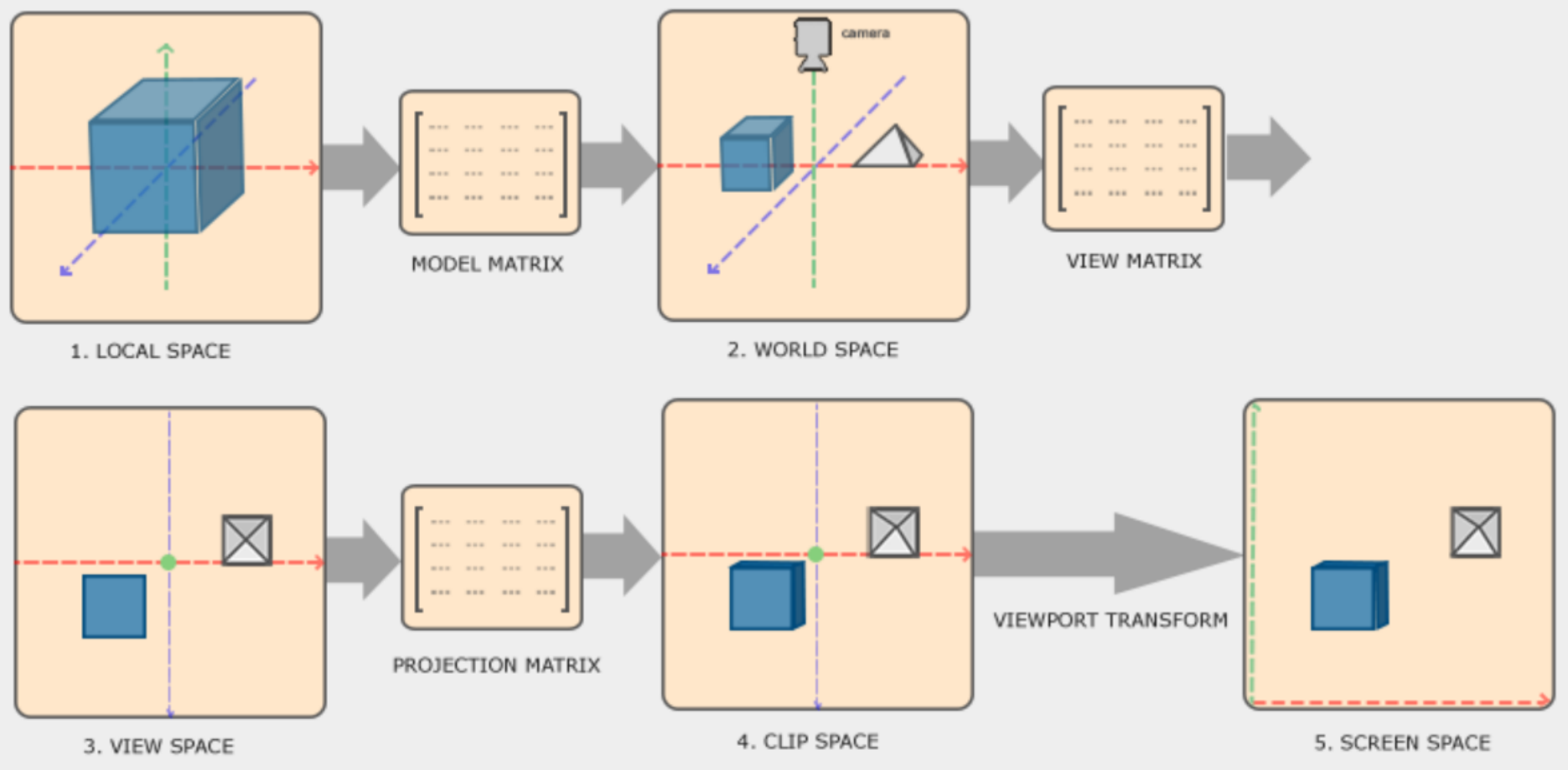
The Rendering Pipeline



Several coordinate systems exist

- Local or Object-space coordinates
  - The coordinates of your object relative to its local origin; these coordinates are the ones you declared or loaded, the values your object begins with.
- Global or World-space coordinates
  - Coordinates relative to the global origin of the whole scene. All objects will be eventually converted to the same world's origin.
- Viewing or Camera coordinates
  - Coordinates as seen from the camera or the viewer's point of view. After the coordinates are in view space they can be projected to the frame buffer.
- Clip coordinates
  - Clip coordinates are obtained when all objects in camera coordinates are transformed to be inside the “viewing frustum” with coordinates normalized in [-1,1].
- Window or Screen-space coordinates
  - Lastly, clip coordinates are transformed to screen coordinates in a process called viewport transformation. It transforms coordinates from [-1,1] to the coordinate range defined by glViewport. The resulting coordinates are then sent to the rasterizer (shaders) to turn them into fragments (pixels to display).

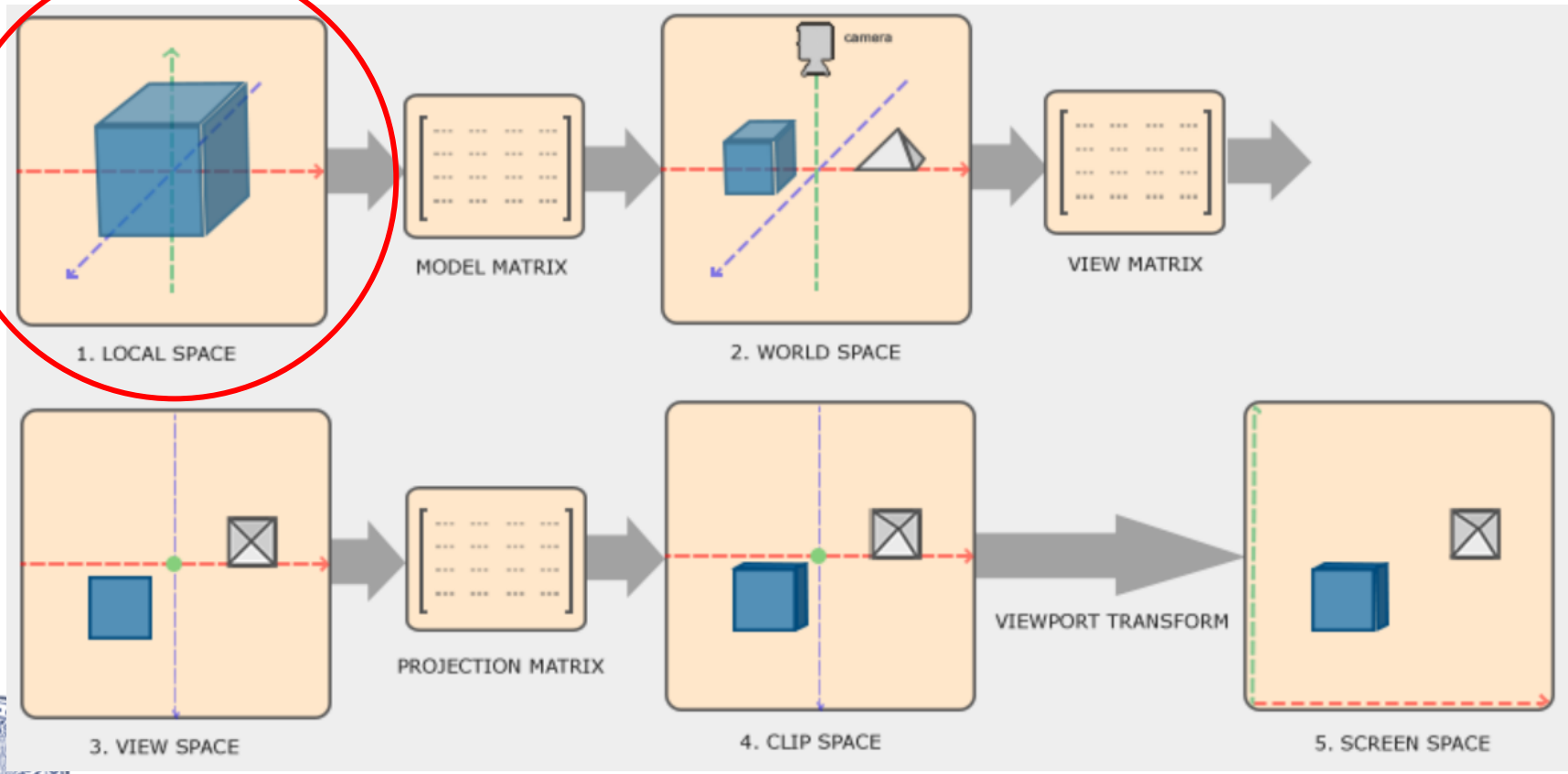
Several coordinate systems exist



https://sciencesoftcode.wordpress.com/2020/09/02/learnopengl/

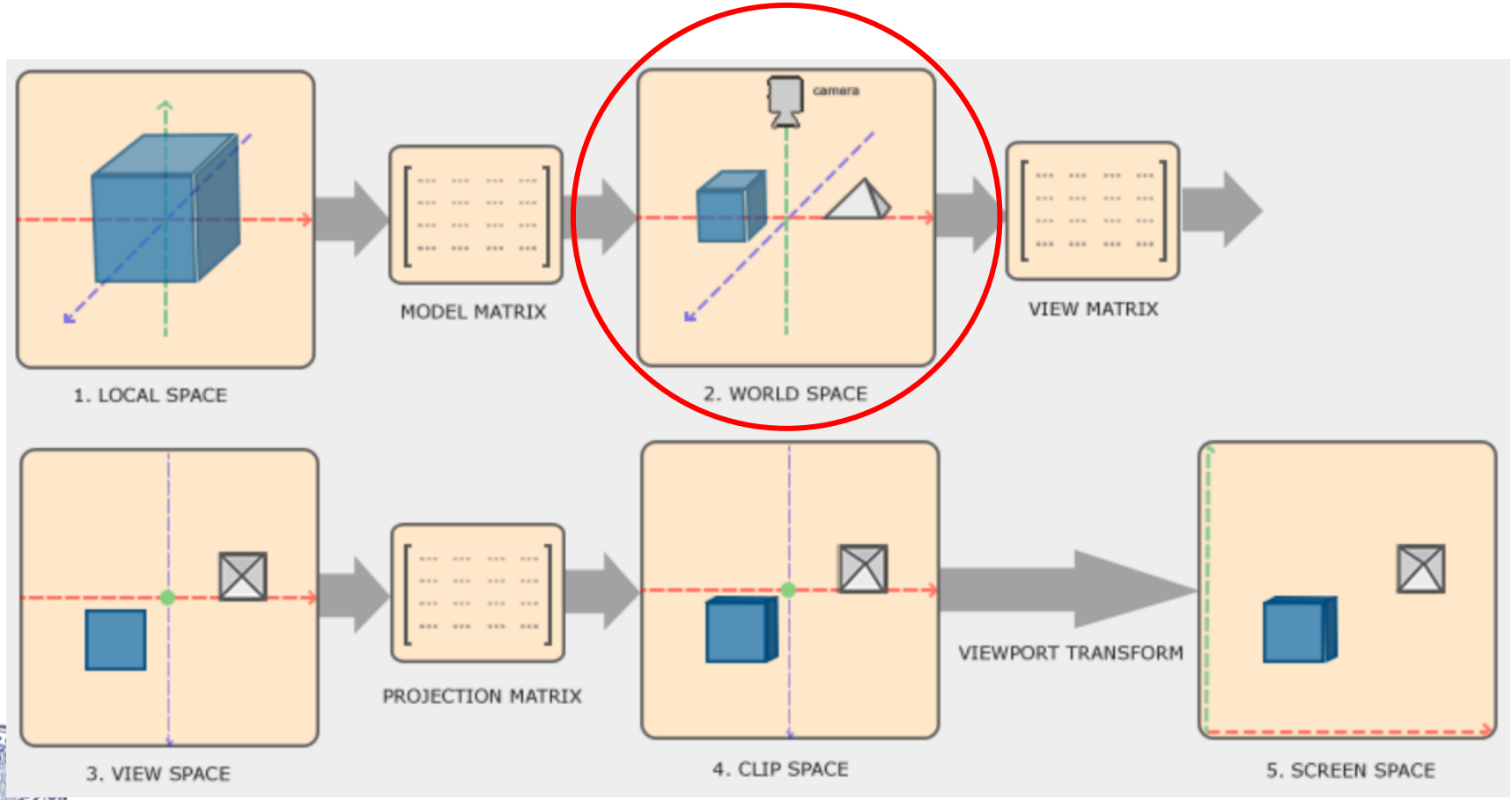
Several coordinate systems exist

- Local or Object-space coordinates
  - The coordinates of your object relative to its local origin; these coordinates are the ones you declared or loaded, the values your object begins with.



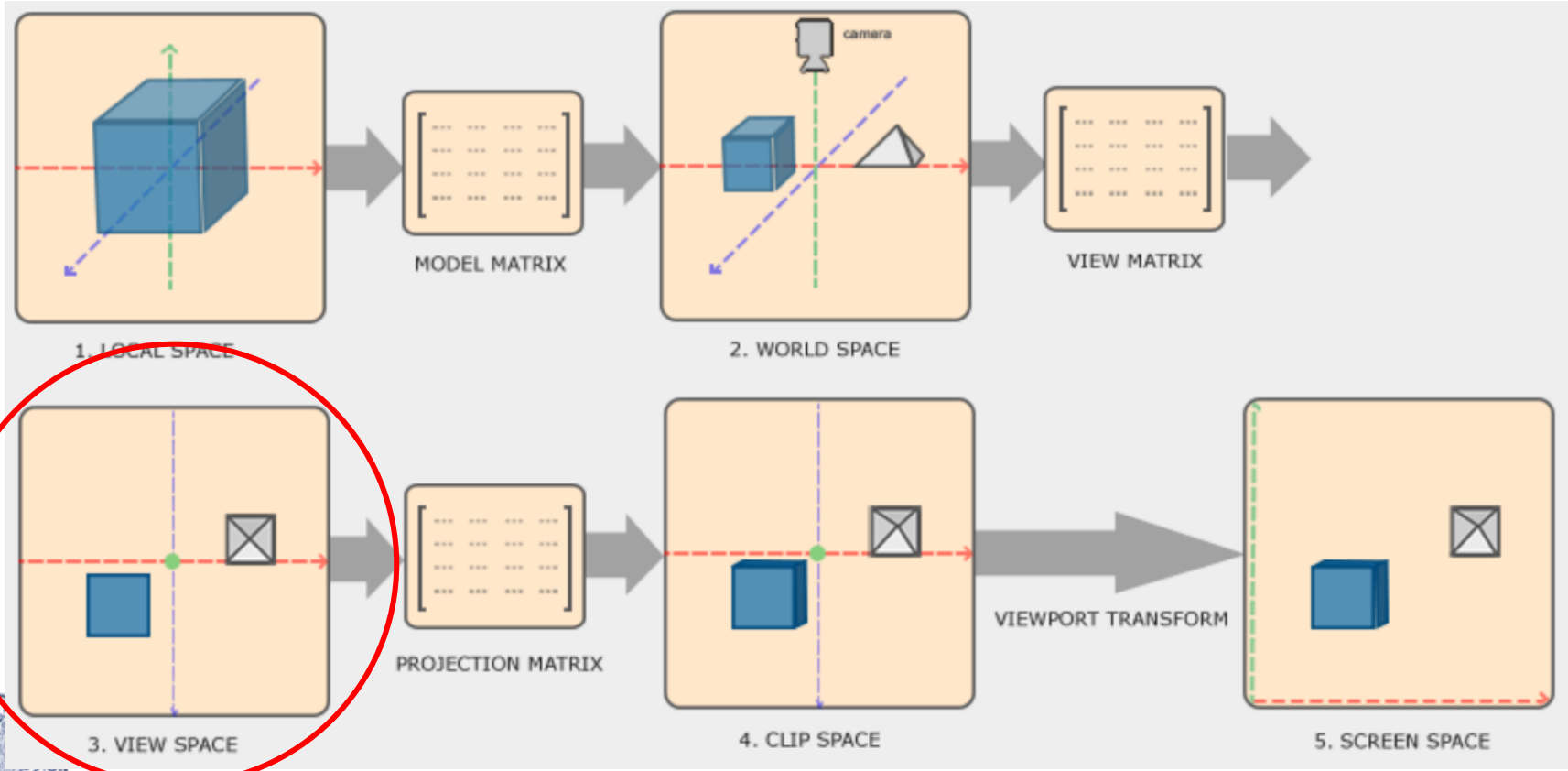
Several coordinate systems exist

- Global or World-space coordinates
  - Coordinates relative to the global origin of the whole scene. All objects will be eventually converted to the same world's origin.



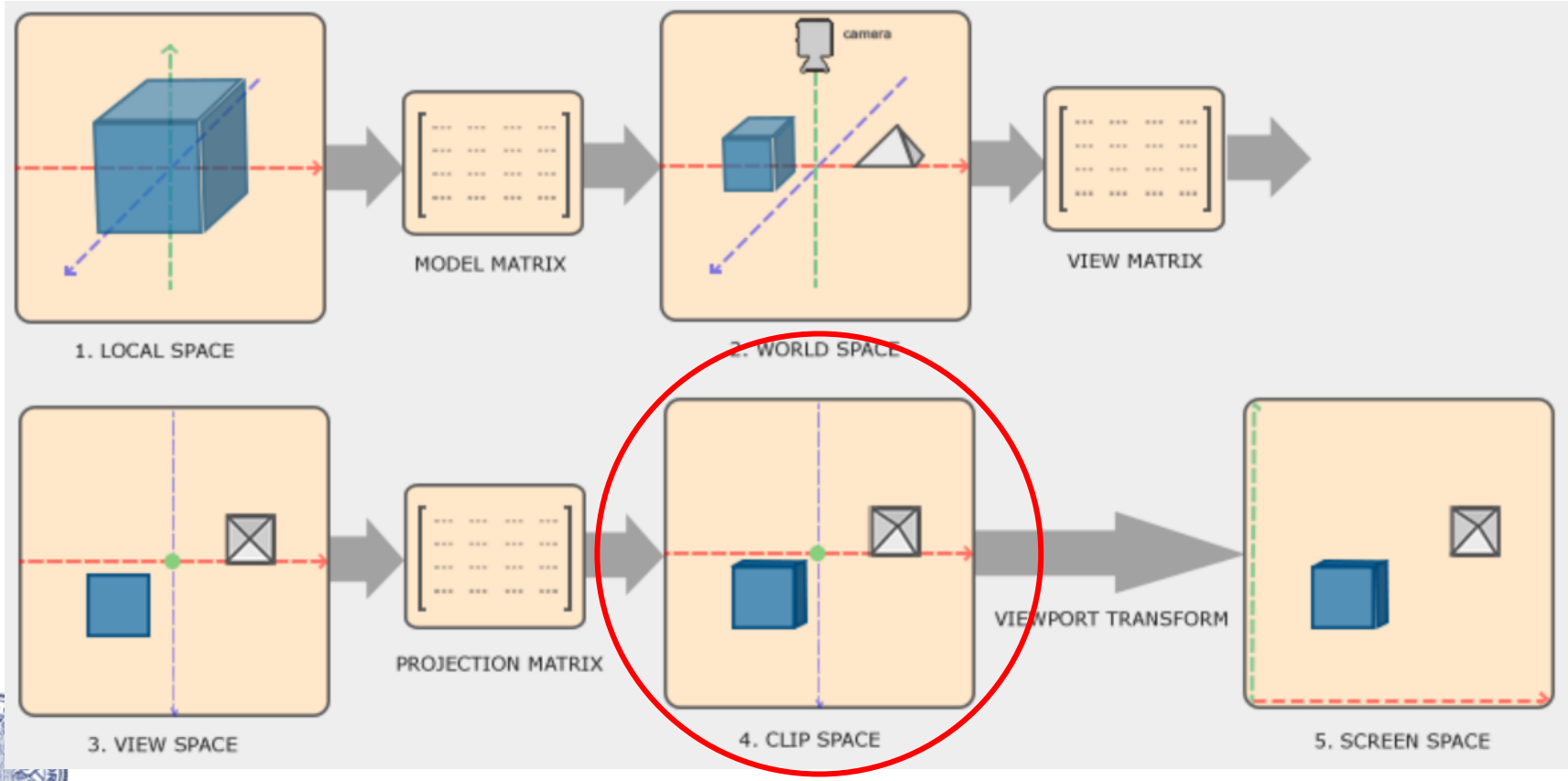
Several coordinate systems exist

- Viewing or Camera coordinates
  - Coordinates as seen from the camera or the viewer's point of view. After the coordinates are in view space they can be projected to the frame buffer.



Several coordinate systems exist

- Clip coordinates
  - Clip coordinates are obtained when all objects in camera coordinates are transformed to be inside the “viewing frustum” with coordinates normalized in [-1,1].



Several coordinate systems exist

- Window or Screen-space coordinates
  - Lastly, clip coordinates are transformed to screen coordinates in a process called viewport transformation. It transforms coordinates from [-1,1] to the coordinate range defined by glViewport. The resulting coordinates are then sent to the rasterizer (shaders) to turn them into fragments (pixels to display).