KTH
VETENSKAP
OCH KONST

ROYAL INSTITUTE
OF TECHNOLOGY

DH2323 DGI17

INTRODUCTION TO
COMPUTER GRAPHICS AND INTERACTION

IMAGE-BASED RENDERING AND ANIMATION

Christopher Peters

CST, KTH Royal Institute of Technology, Sweden

chpeters@kth.se

http://kth.academia.edu/ChristopherEdwardPeters

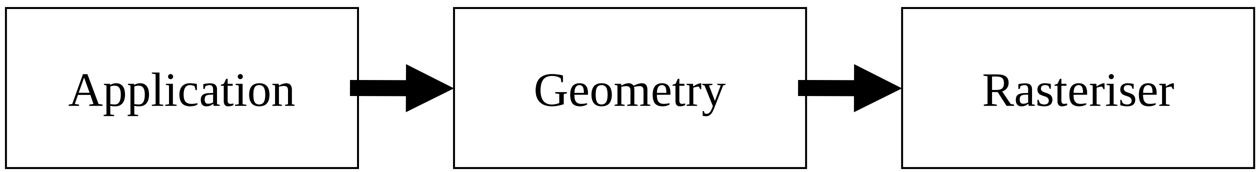
Graphics Pipeline Architecture

Can divide pipeline into three conceptual stages:

Application (input, animations, think SDL)

Geometry (transforms, projections, lighting)

Rasteriser (draw image as pixels)



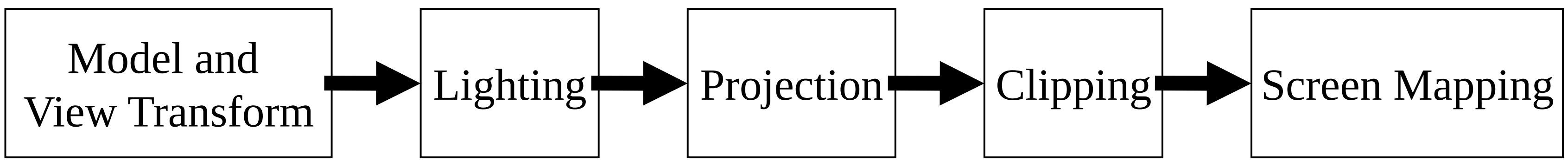
These define the core structure of the pipeline

Geometry Stage

Responsible for polygon and vertex operations

Consists of five sub-stages:

- Model and View Transform
- Lighting and Shading
- Projection
- Clipping
- Screen Mapping



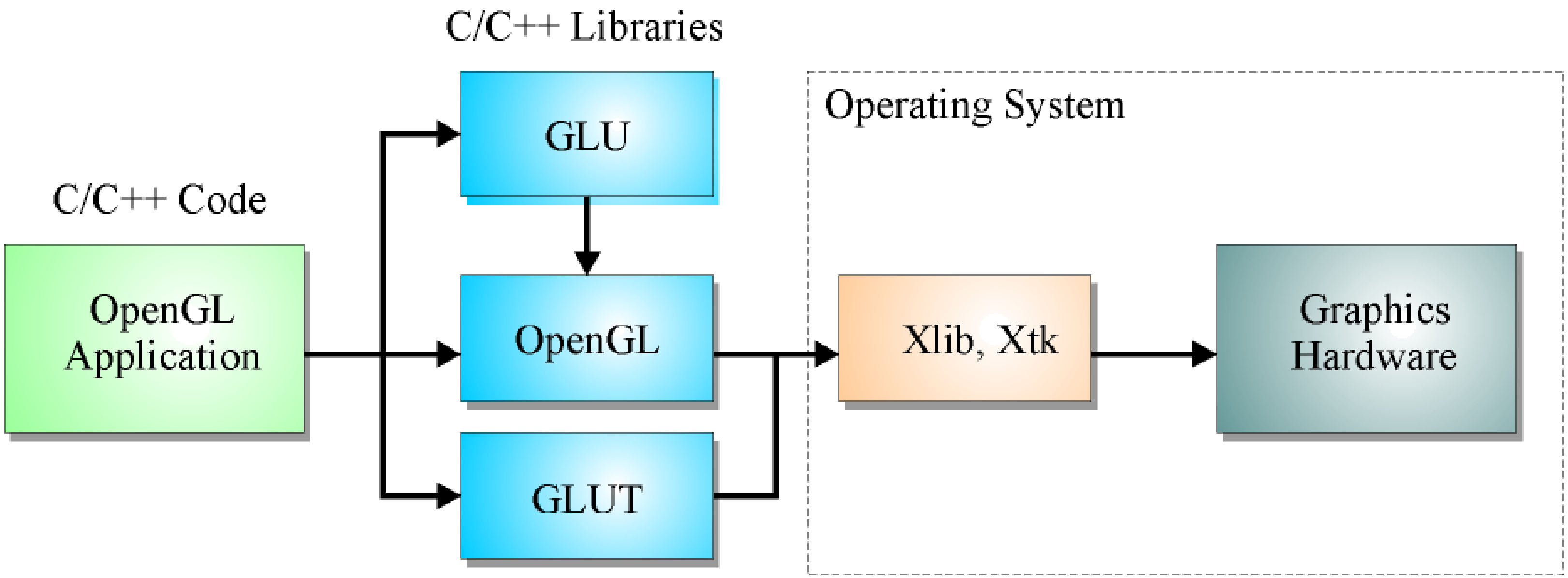
What is OpenGL ?

Software interface to graphics hardware

Commands for interactive three-dimensional graphics

Hardware independent interface

Drawing operations performed by underlying system and hardware



I.

Image-based Rendering

Image-based Rendering

X constitutes:

- Geometry
- Texture
- Lighting
- Material

Parameterisation of the 'world'

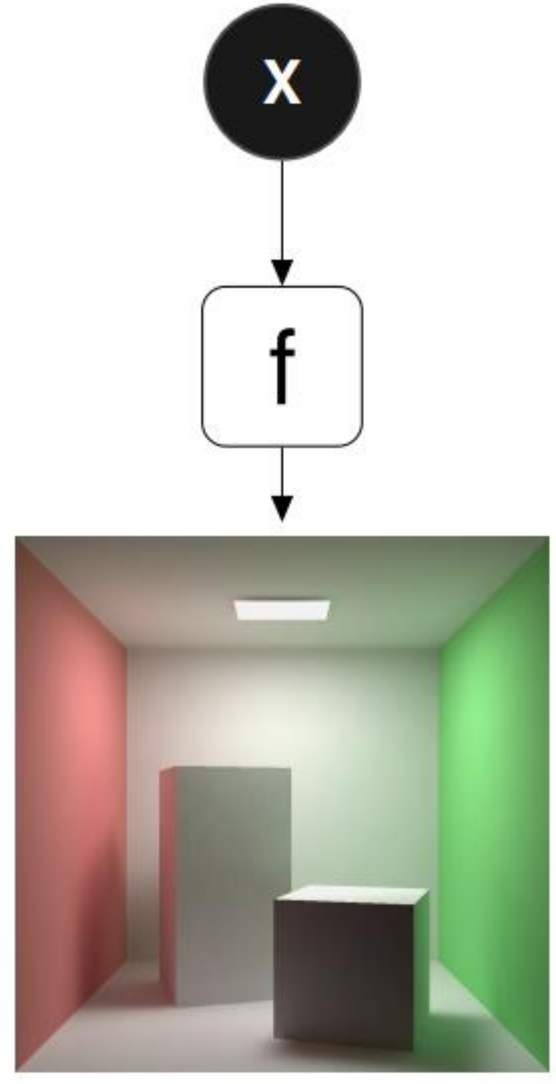


Image-based Rendering

How does *f(.)* manipulate **X**?

- Light
- Surface interaction
- Light transport

Formulate model *f(.)* through assumptions

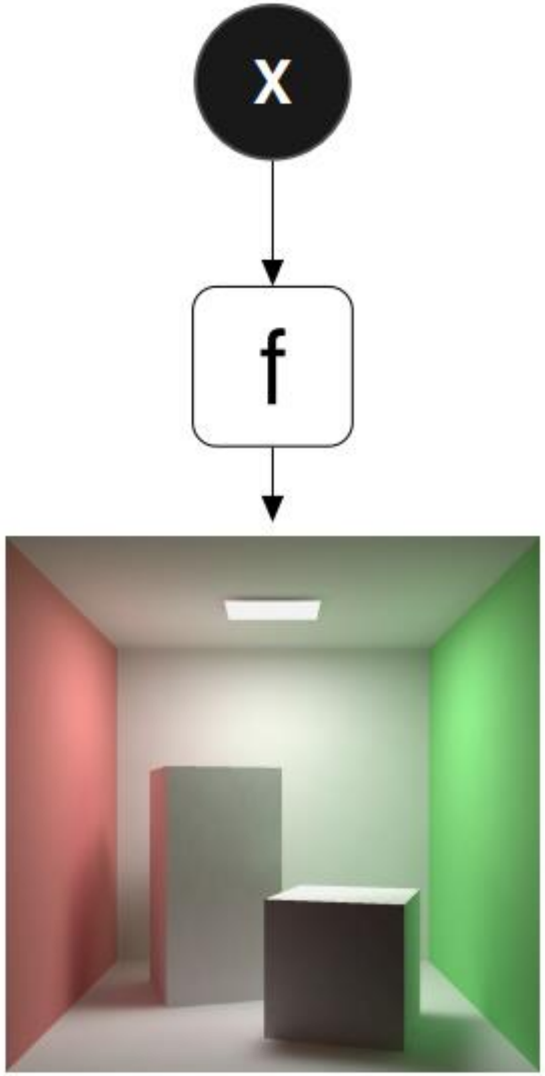


Image-based Rendering

Computer Vision
- Image easy to acquire
- Solve inverse problem
- What parameters have generated the image?

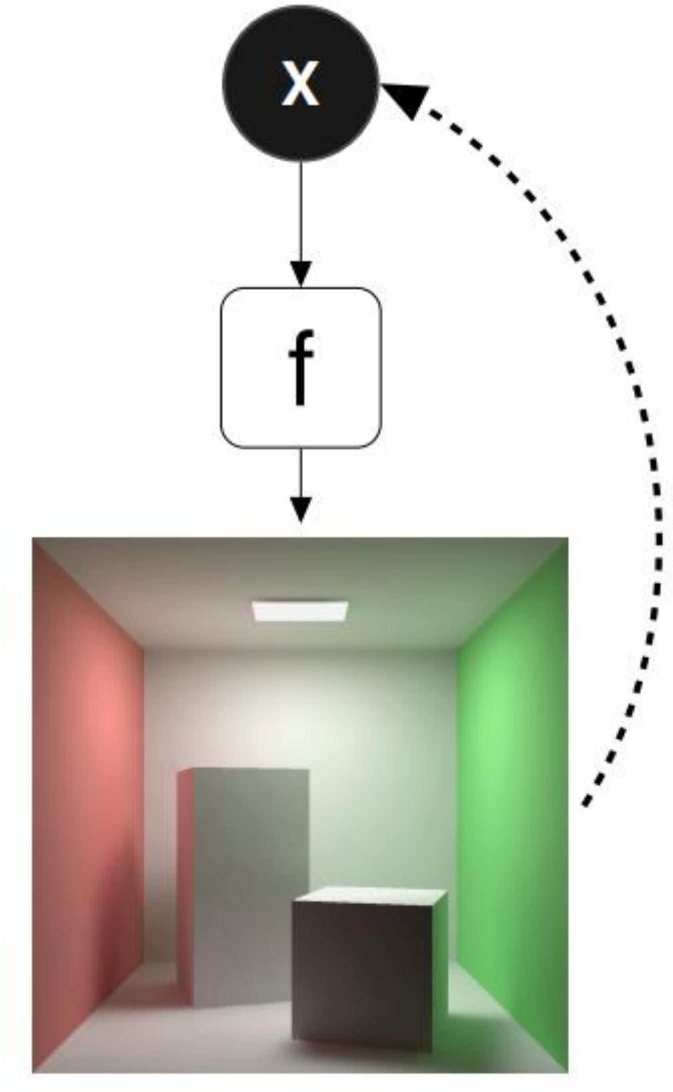


Image-based Rendering

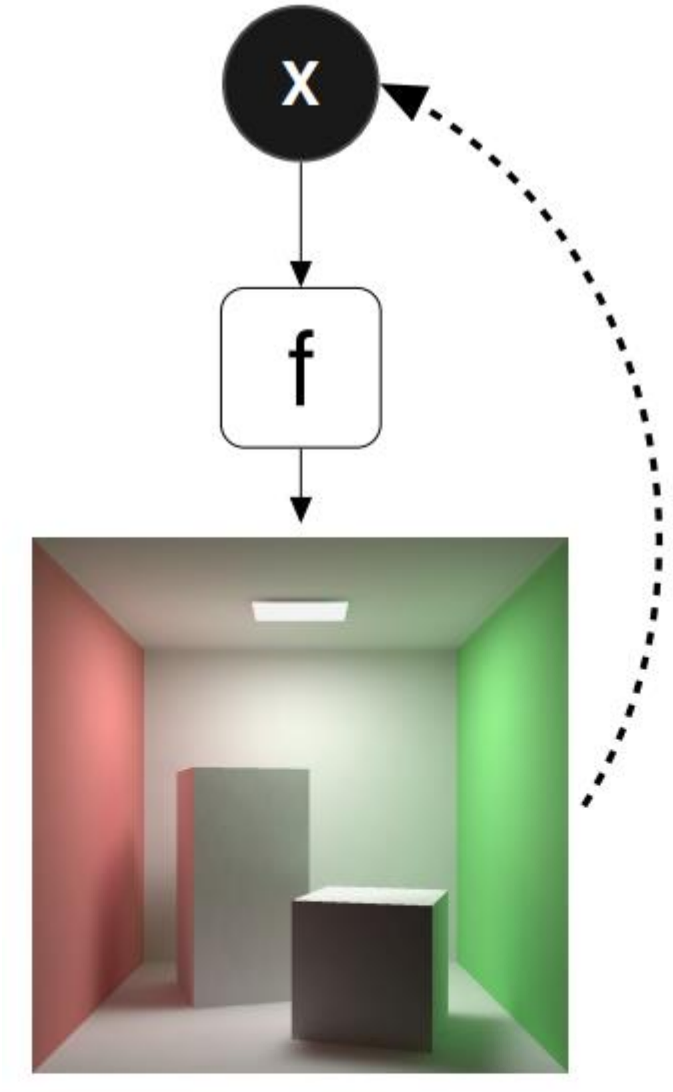
Challenge:

Given an image recover the parameters

- Texture
- Light
- Geometry

Models still valid

- $x_i = f^{-1}(y_i)$



ROYAL INSTITUTE OF TECHNOLOGY

Image-based Rendering

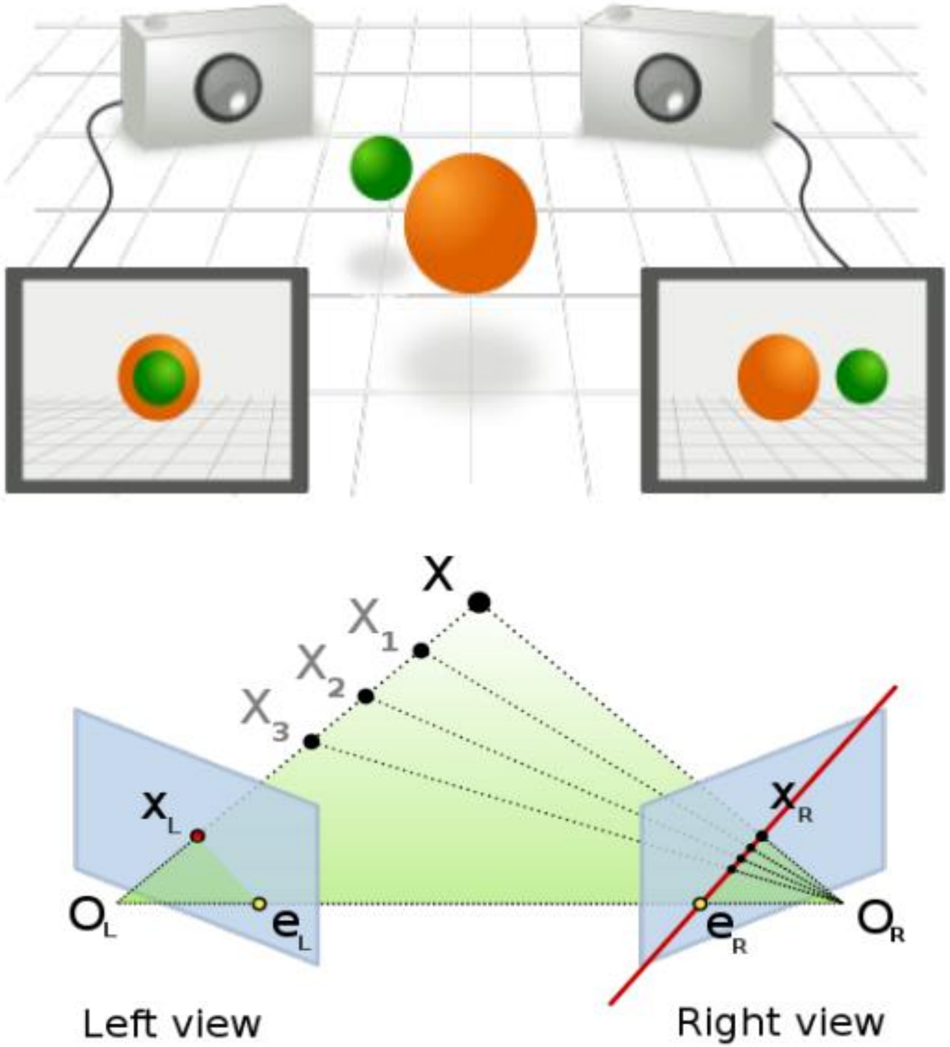
Rendering: generate images from viewpoints

Image-based rendering: replace geometry and material attributes with real images

Most realistic image? A photograph

- Lacks flexibility
- Cannot change lighting or viewpoint
- Combine images to produce a new one

Epipolar Geometry



- The geometry of stereo vision
- Two cameras viewing 3D scene from different positions
- Study geometric relations between 3D points and their 2D projections
- More images = more constraints

II.

Animation

ROYAL INSTITUTE OF TECHNOLOGY

Traditional Animation

Showing consecutive related static images one after another produces the perception of a moving image

Master artists draw certain important **key-frames** in the animation

Apprentices draw the multitude of frames in-between these key-frames

Called **tweens**

Computer Animation

Objects have an initial configuration and a final configuration (often specified by the artist)

- Position, orientation, etc

Computer calculates the intermediate configurations: *interpolation*

Orientation Interpolation:

Given two key-frame orientations, calculate an intermediate orientation between them

ROYAL INSTITUTE OF TECHNOLOGY

Question

How do we interpolate orientations?

Question

How do we interpolate orientations?

Remember how they are represented

1) Rotations around axes

2) Rotation matrices

Question

How do we interpolate orientations?

Remember how they are represented

1) Rotations around axes

2) Rotation matrices

Possible solutions: Euler angles and rotation matrix interpolation

#1: Euler Angles

An Euler angle is a rotation around a single axis

Any orientation can be specified by composing three rotations

- Each rotation is around one of the **principle axes**
- i.e. (x, y, z) – first rotate around x, then y, then z
- Think of roll, pitch and yaw of a flight simulator

When rotating about a single axis, is possible to interpolate a single value

- However, for more than one axis, interpolating individual angles will not work well
- Unpredictable intermediate rotations
- Gimbal lock

#2: Rotation Matrices

Interpolating between two rotation matrices does not result in a rotation matrix

- Does not preserve rigidity of angles and lengths
- This result of an interpolation of 0.5 between the identity matrix and 90 degrees around the x-axis does not produce a valid rotation matrix:

$$\text{Interpolate}\left(\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}, \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{vmatrix} \right) \equiv \xcancel{\begin{vmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & -0.5 & 0.5 \end{vmatrix}}$$

Solution

Use *quaternion interpolation*

Quaternions don't suffer from Gimbal lock

Can be represented as 4 numbers instead of 9 of a 3x3 matrix

Trivial conversion between angle/axis representation

Interpolation between two quaternions is easy (once you know how)

Quaternion looks like this:

$q[w,(x,y,z)]$ also written $q[w,v]$ where $v = (x,y,z)$

$q = w + xi + yj + zk$

Representation

For a right-hand rotation of θ **radians** about unit vector *v*, quaternion is:

$q = (\cos(\theta/2); \mathbf{v} \sin(\theta/2))$

- Note how the 3 imaginary coordinates are noted as a vector
- Only **unit quaternions** represent rotations
  - Such a quaternion describes a point on the 4D unit hyper-sphere
- Important note: q and –q represent the **exact same** orientation
- Different methods for doing quaternion interpolation:

LERP, <u>SLERP</u> (Spherical linear interpolation)

In Practice

- Not always the best choice
  - Quaternions are (as you will have noticed) hard to visualise and think about
  - If another method will do and is simpler, it will be a more appropriate choice
- But…
  - Extremely useful in many situations where other representations are awkward
  - Easy to use in your own programs once you have a quaternion class
  - See Animation track labs and GLM library

Reminders

- You should be working on Lab 3
- Labs due on Friday 26th May
  – Canvas DH2323 submission will open this week

- Project work (due May 31st)

- Lab session:

  Today from 13:00-15:00,

  Visualization Studio

Next lecture

- User studies and perception
- Will take place in B2
- Monday 15th May
- 10:00 – 12:00