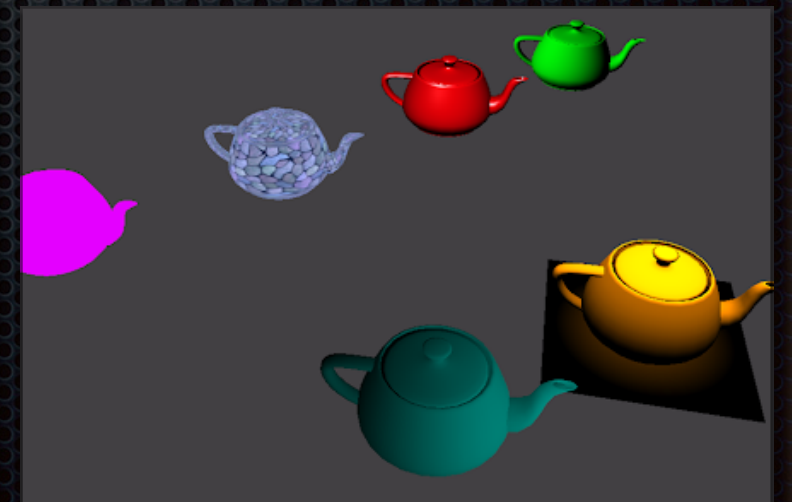
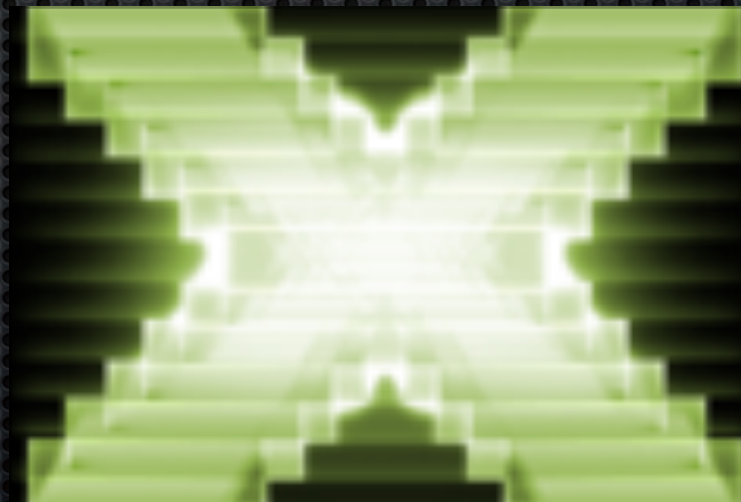


Graphic System & Shader Programming

David Hackbarth



A Short History

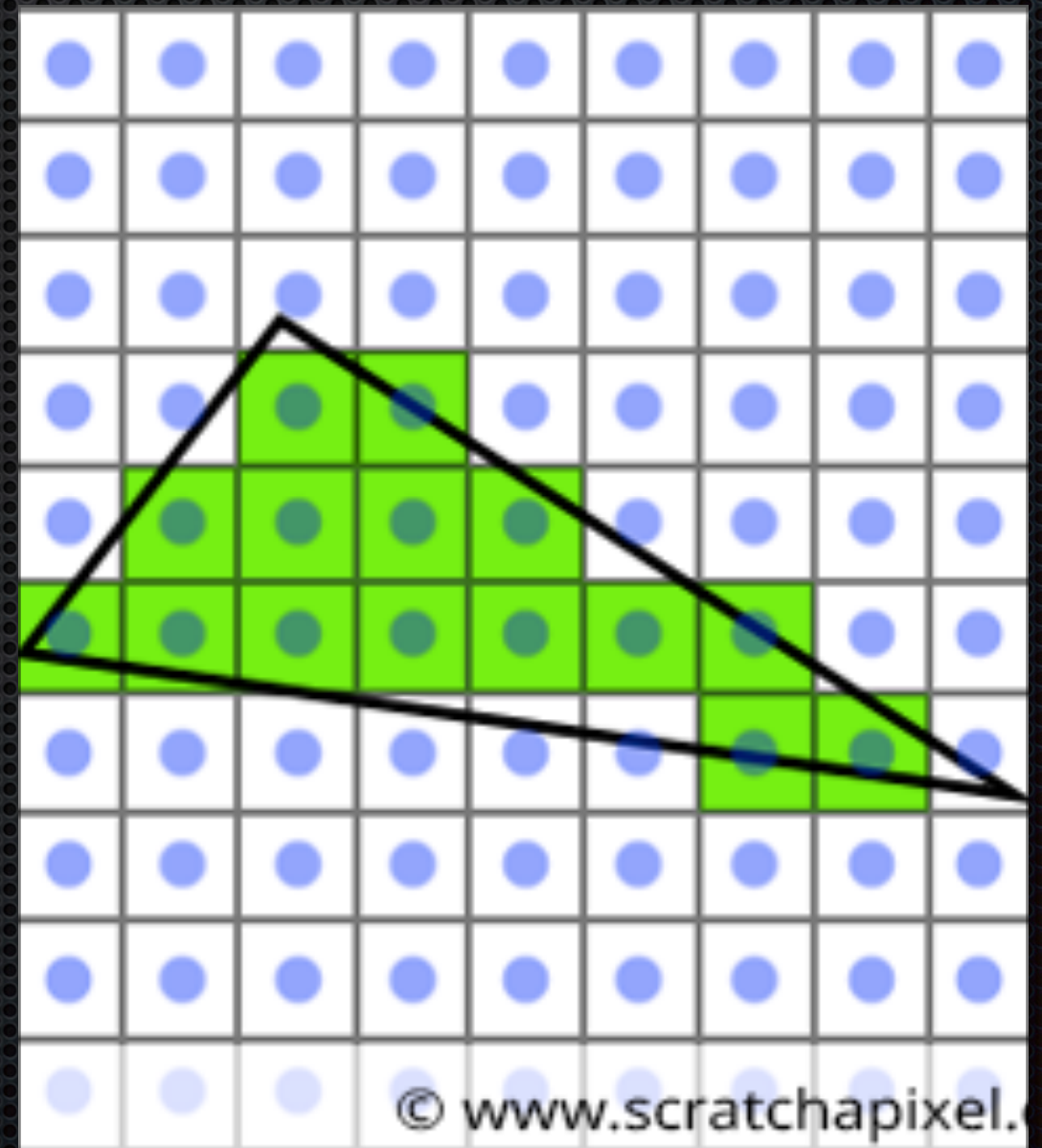
Vector graphics

- ✦ 1958 - Tennis For Two (William Higinbotham)
- ✦ 1963 - Sketchpad (Ivan Sutherland)
- ✦ inflexible



Raster graphics

- ✦ 1970s
- ✦ single pixel coloring
- ✦ frame buffer for presentation
- ✦ very slow



Hardware Sprites



- ✦ 1970s
- ✦ limited size
- ✦ limited colors
- ✦ limited amount on screen



Fixed-Function Pipeline

- ✦ 1990s
- ✦ render pipeline is fixed
- ✦ changeable by render states
- ✦ one primitive - one draw call



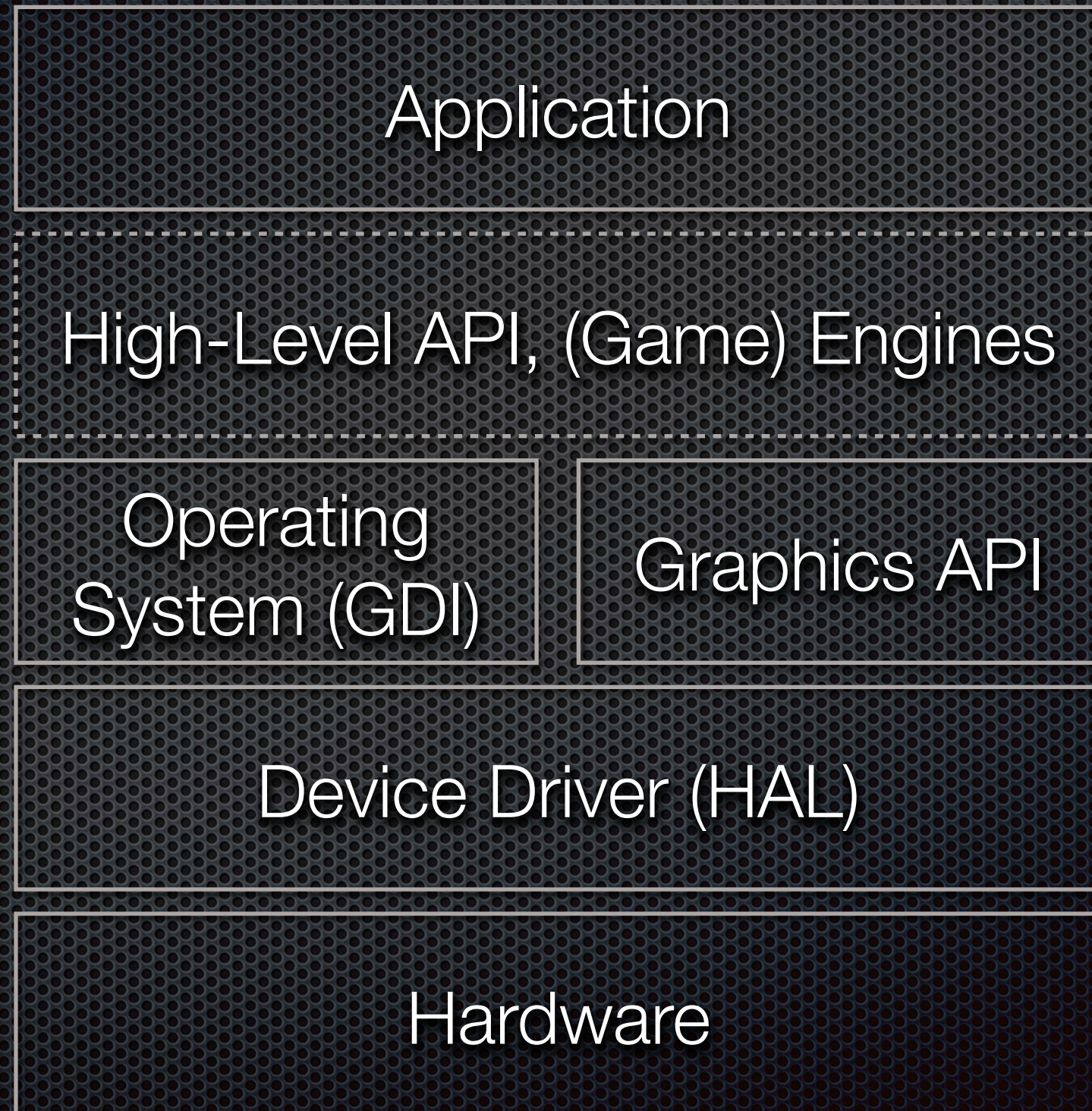
Programmable Pipeline

- ✦ since 2001
- ✦ a part of render pipeline is fixed
- ✦ changeable by render states
- ✦ some parts are programmable
- ✦ many primitives - one draw call

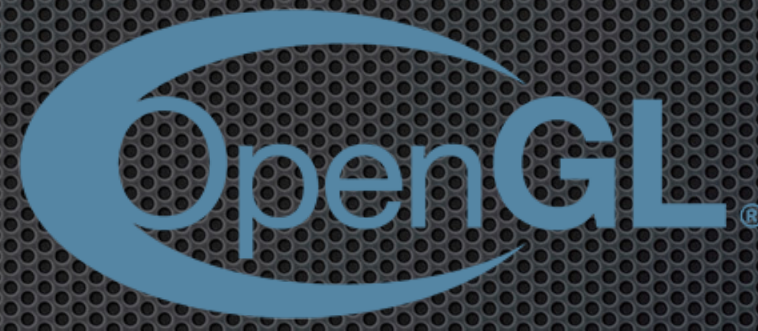


Graphics API

Abstract Layer



Overview

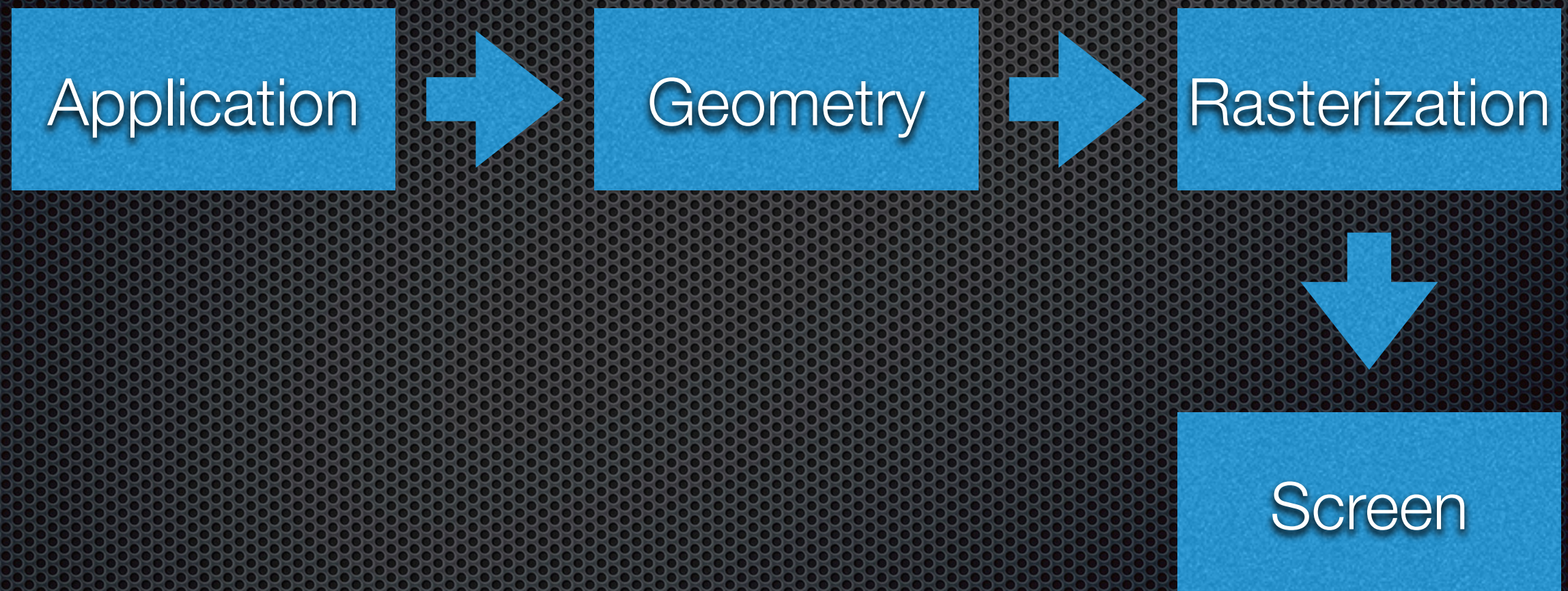


DirectX

- ✦ group of many libraries
 - ✦ Direct3D
 - ✦ DirectSound
 - ✦ DirectInput (new XInput)
 - ✦ Direct2D (removed)
 - ✦ etc.
- ✦ Version: 12
- ✦ too complex for beginners
- ✦ instead 11

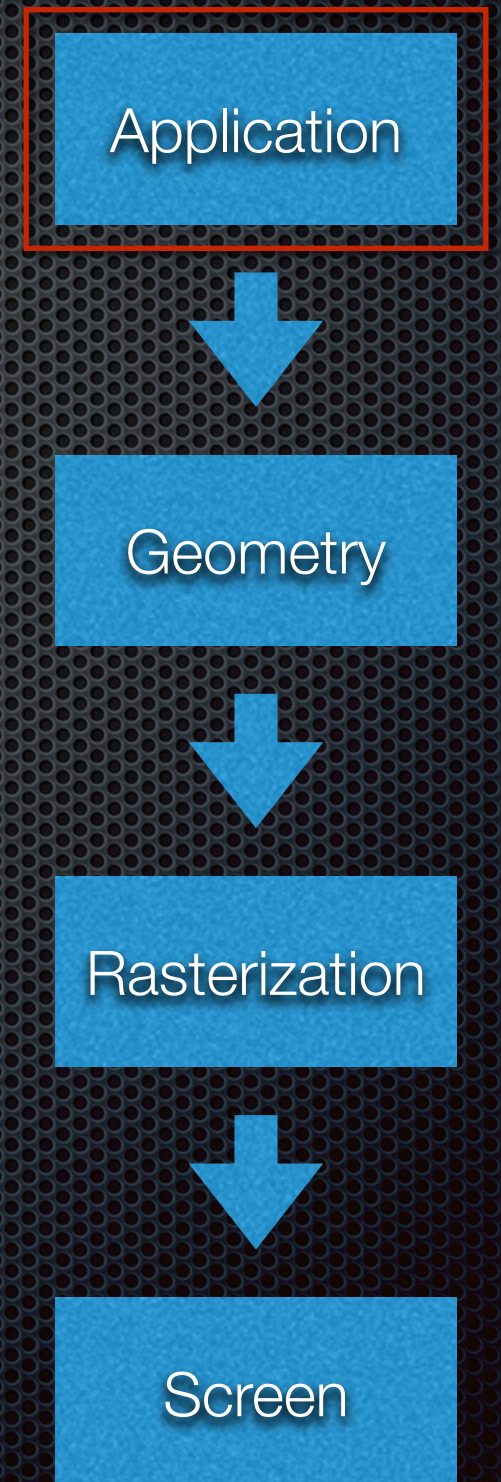
Programmable Pipeline

Structure



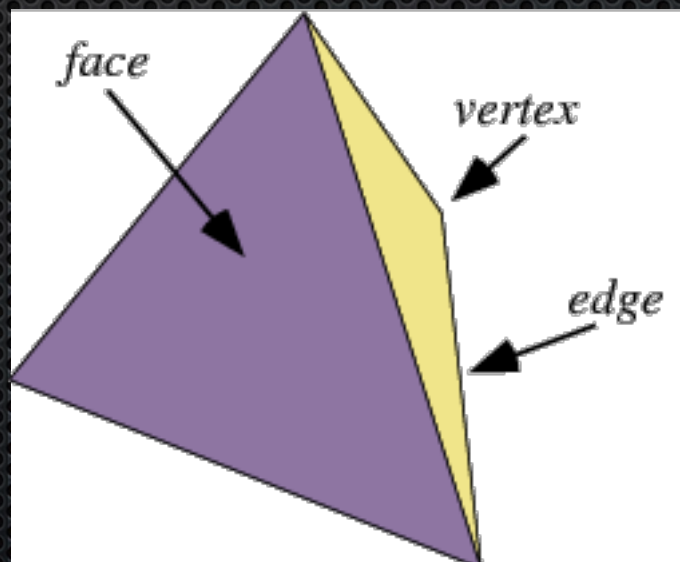
Application

- execution on CPU
 - create models, no rendering
 - gameplay
 - physics
 - input
 - etc.
- scene graph as octree



Vertex

- structure for representing a point in space
- consists of at least a position in space (normally 3D)
- may have additional attributes



Application



Geometry



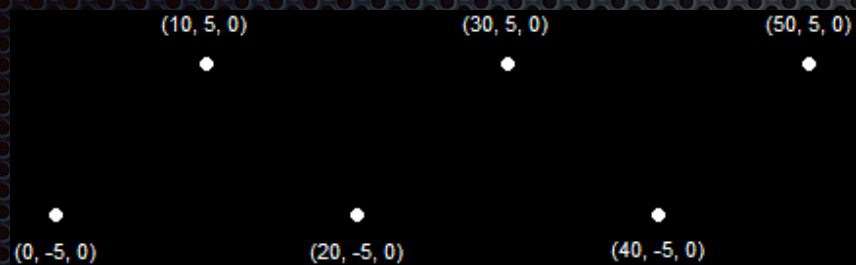
Rasterization



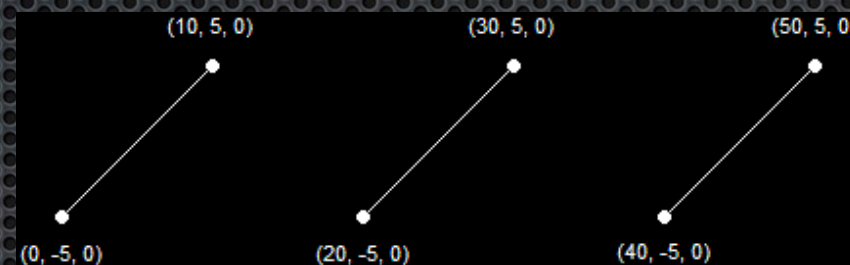
Screen

Primitives

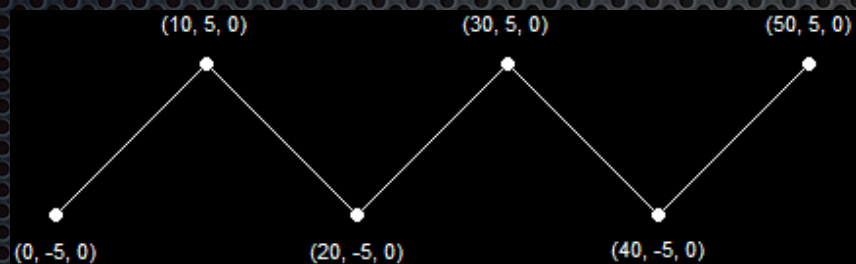
Pointlist



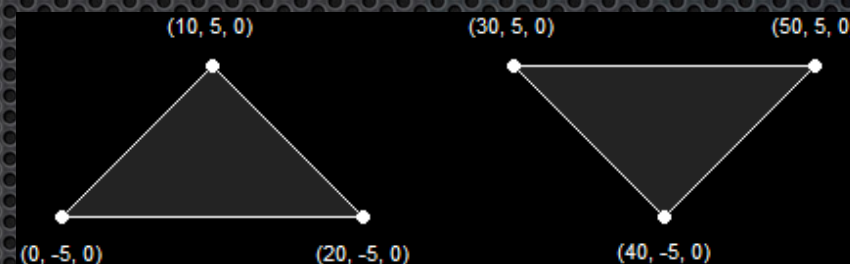
Linelist



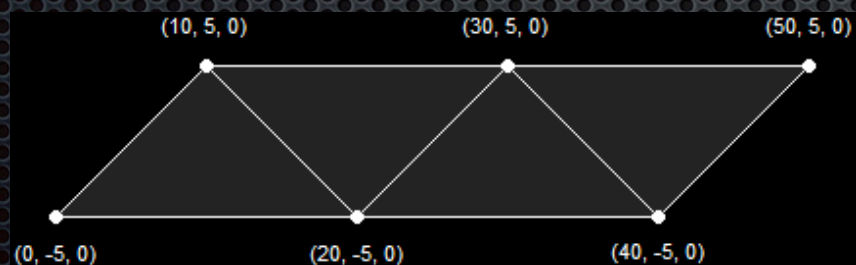
Linestrip



Trianglelist



Trianglestrip



Application



Geometry



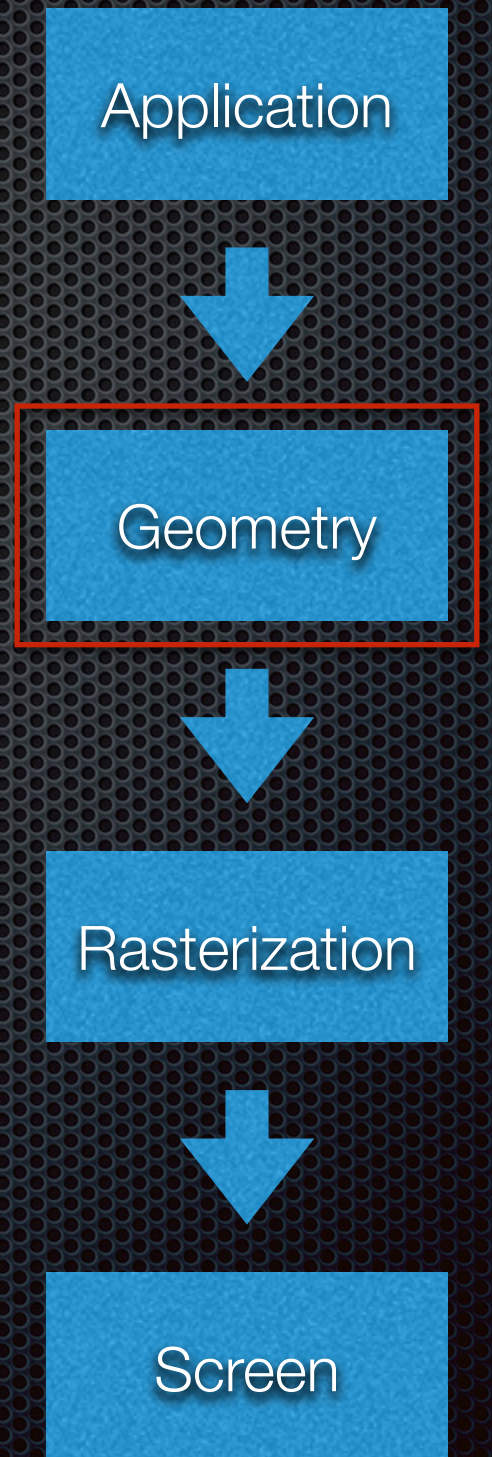
Rasterization



Screen

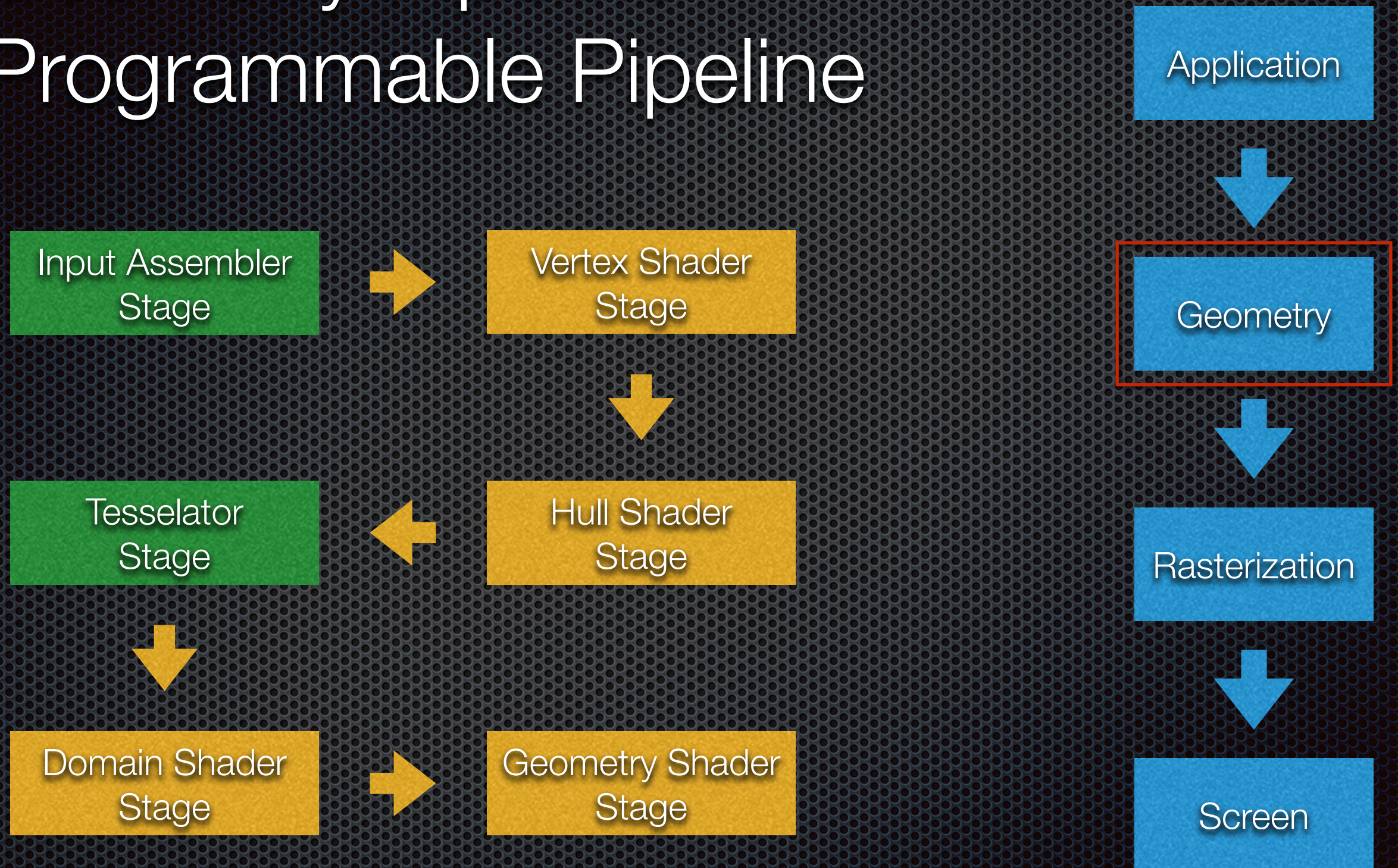
Geometry

- ✦ transformation
- ✦ tessellation
- ✦ projection
- ✦ clipping
- ✦ additional geometry



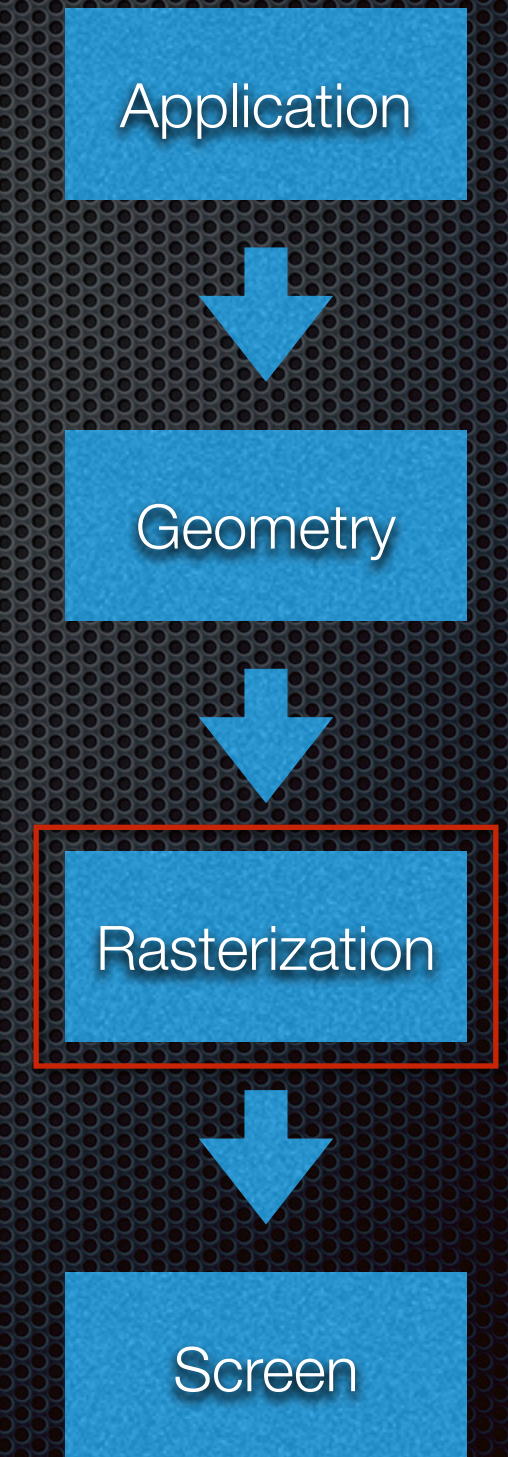
Geometry Pipeline

Programmable Pipeline



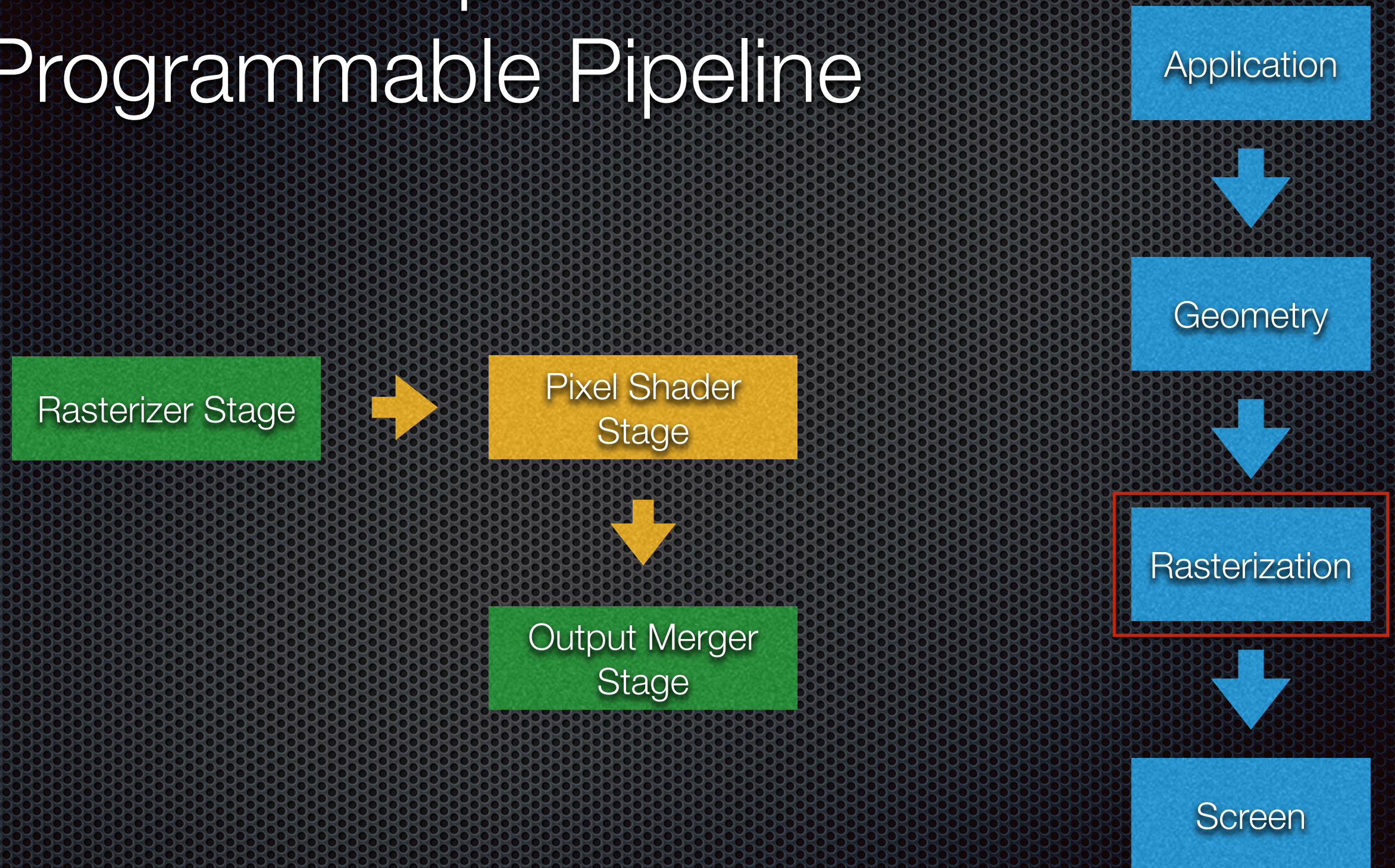
Rasterization

- ✦ primitives rasterized
- ✦ pixels (fragments) created/colored
- ✦ final stage before screen



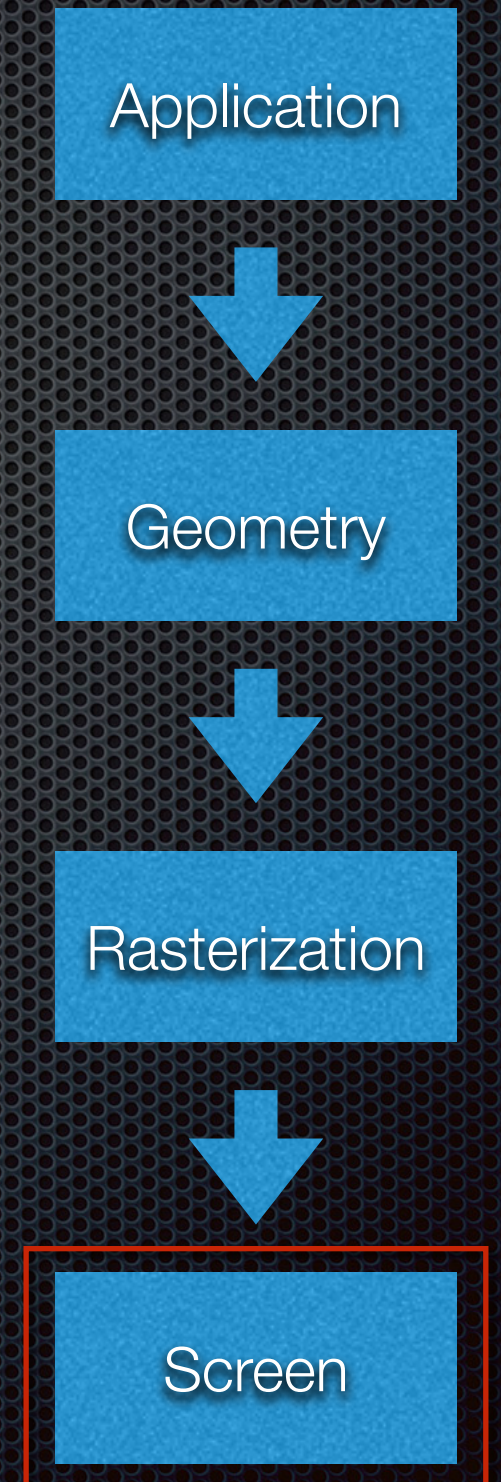
Rasterizer Pipeline

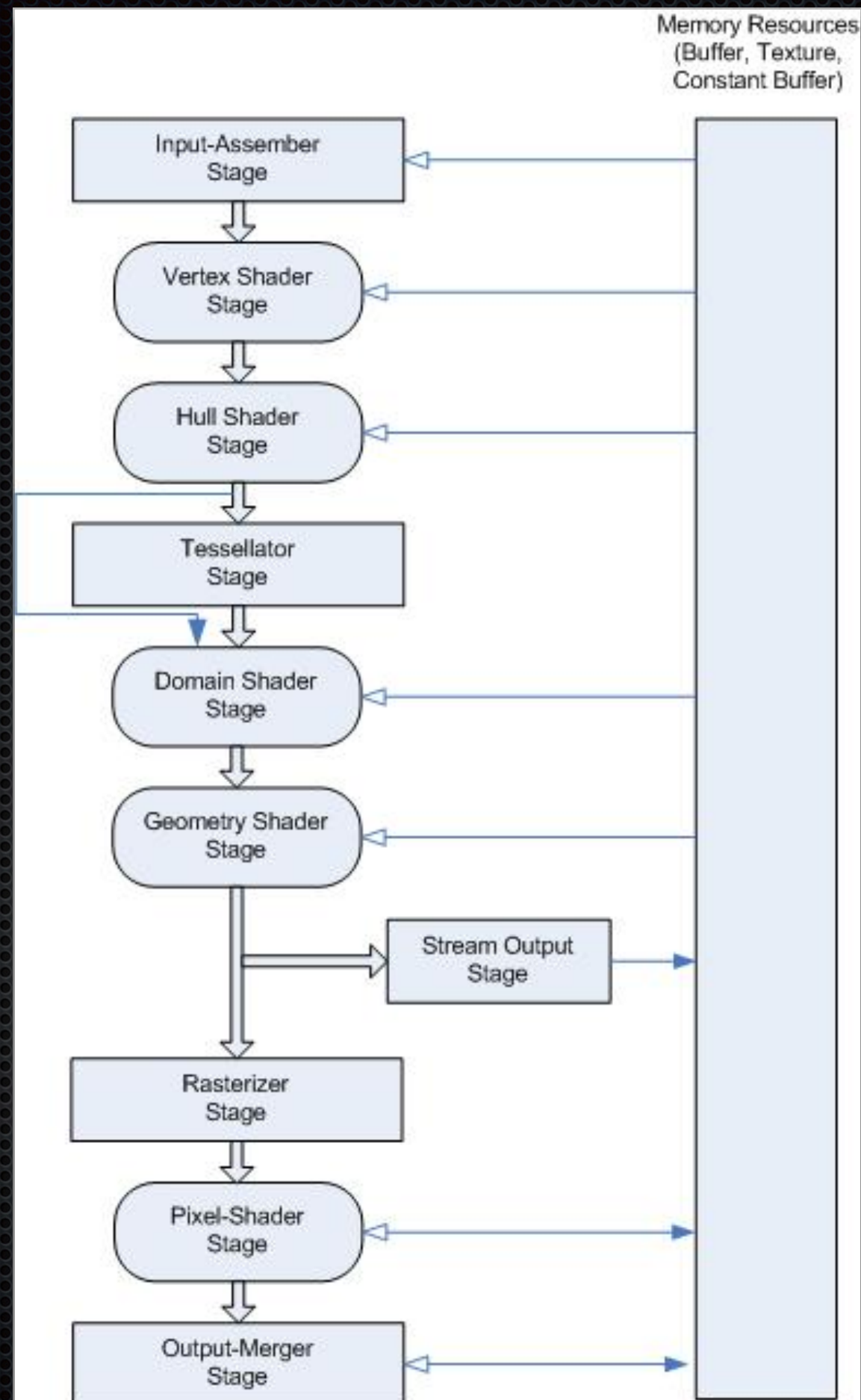
Programmable Pipeline



Screen

- ✧ frame presentation
- ✧ Swapchain
 - ✧ Frontbuffer
 - ✧ Backbuffer
- ✧ Vertical Synchronisation





Implementation

Window

- ✦ every application needs a window
- ✦ graphic accelerated content needs also a window
- ✦ window will be created with the help of Windows-API
- ✦ communication between application and Windows with the help of messages

Coding Time



Vertexbuffer

- vertices will be saved in a vertexbuffer
- each vertex will be saved sequentially
- each vertex is an instance of a struct
- consists all informations, e.g.
 - position
 - normal
 - uv

Indexbuffer

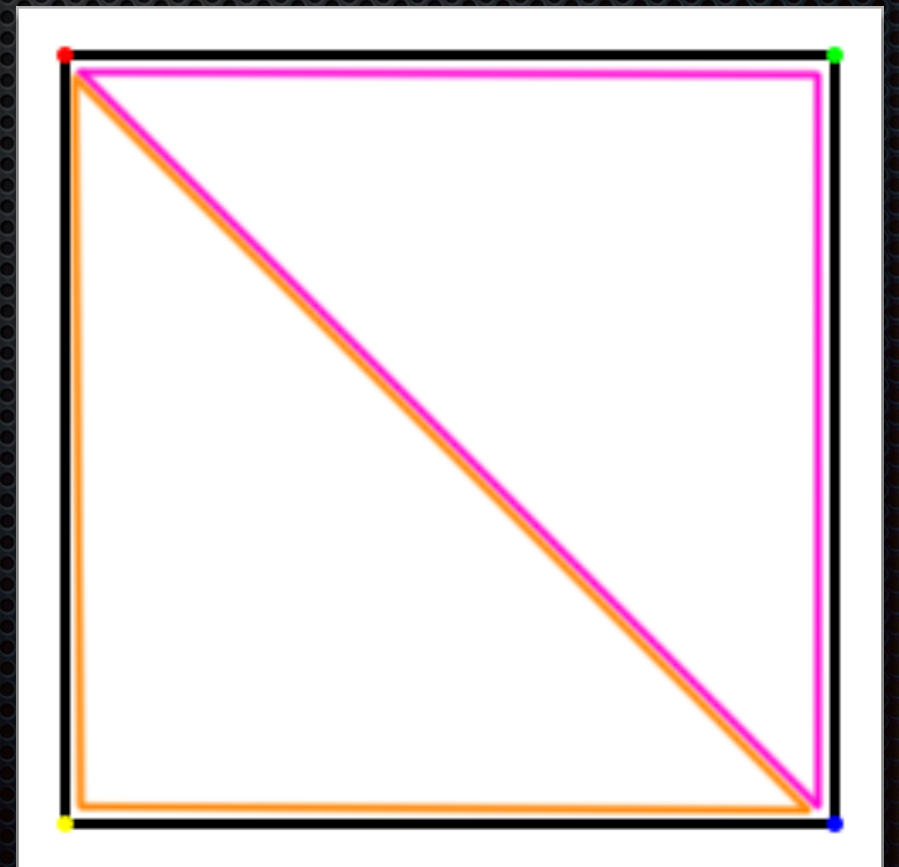
- indices refer to one vertex in vertexbuffer
- indices can refer to a vertex more than once, but
- only once for one primitive
- indices are saved sequentially
- primitives are
 - triangle
 - line
 - point

Create a Mesh

VertexBuffer



IndexBuffer



Coding Time

There are two types of people.

```
if (Condition)
{
    Statements
    /*
     *
     */
}
```

```
if (Condition) {
    Statements
    /*
     *
     */
}
```

Programmers will know.

Shader

- ✦ a shader is a program run on a gpu
- ✦ optimized for parallel execution
- ✦ different shaders for different stages
- ✦ highly flexible instead of fixed-function pipeline

Shader Languages

- ✦ High Level Shading Language (HLSL)
- ✦ OpenGL Shading Language (GLSL)
- ✦ C for Graphics Effects (CgFx)

Kind of Shaders

- ✦ Vertex Shader
- ✦ Pixel Shader
- ✦ Geometry Shader (since Version 4)
- ✦ Hull Shader (since Version 5)
- ✦ Domain Shader (since Version 5)
- ✦ Compute Shader

Shader Resources

- ✦ set for every shader individually
- ✦ Textures
- ✦ Samplers
- ✦ Constant Buffers

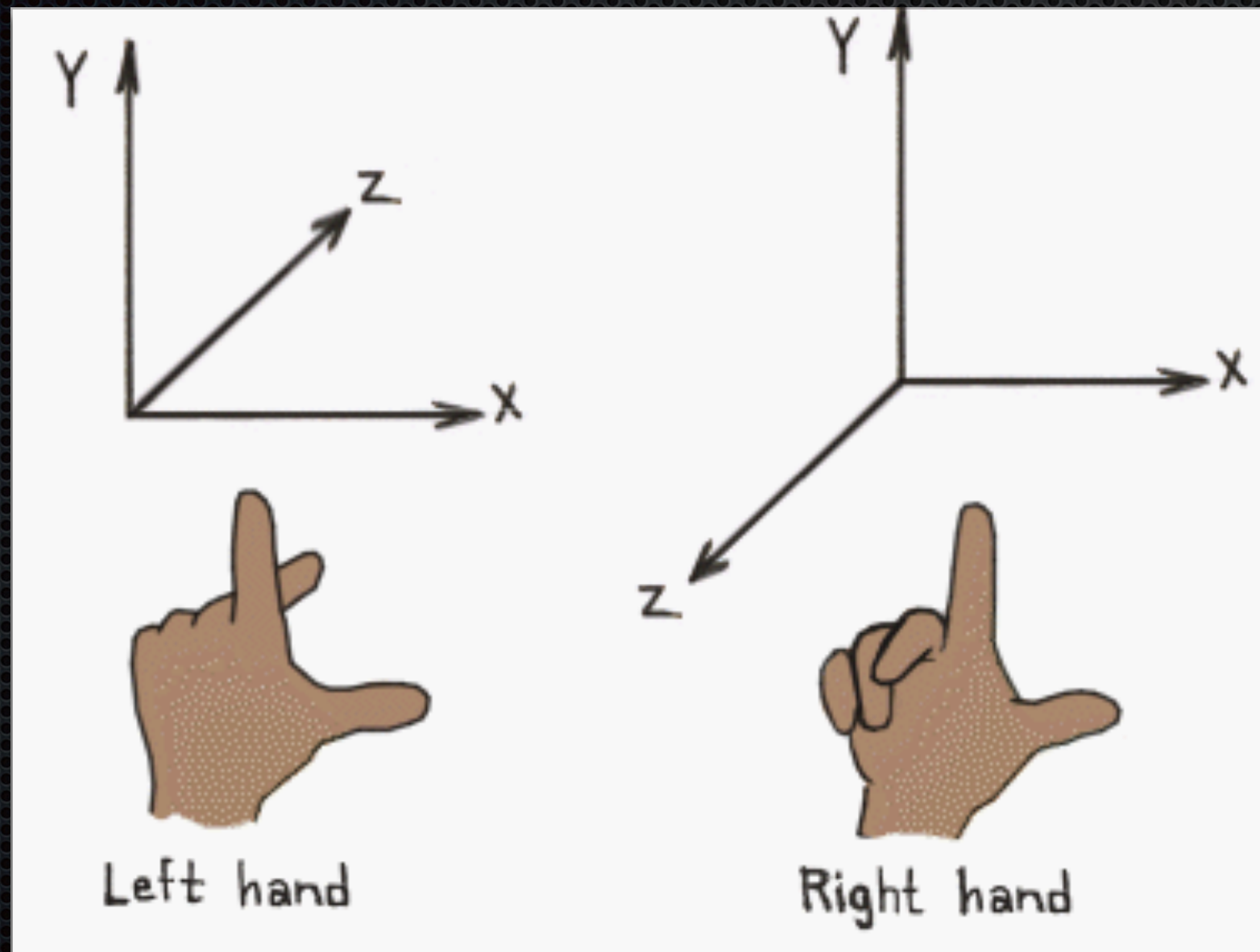
Worldtransformation

- transform an object from local into world space
- we are creating a world/scene
- consists of
 - Translation
 - Rotation
 - Scaling
- normal order is
scaling * rotation (x * y * z) * translation

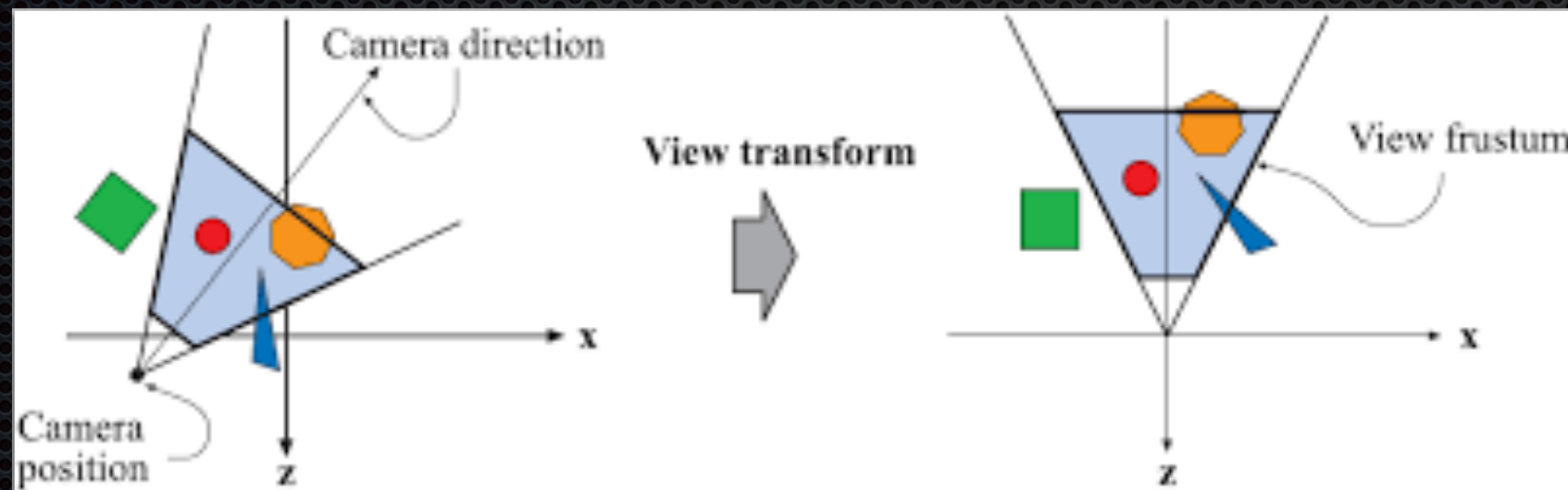
Viewtransformation

- ✦ transform all object into camera space
- ✦ original of this coordinate system is camera position
- ✦ rotation of this coordinate system is orientation of camera
- ✦ distinguish between left and right handed coordinate system

Left & Right Handed Coordinate System



Viewtransformation



Projection

- ✦ project all object onto a 2d plane
for rasterise a picture
- ✦ normalized device coordinates $[-1,+1]$
- ✦ two projection
 - ✦ perspective
 - ✦ orthographic

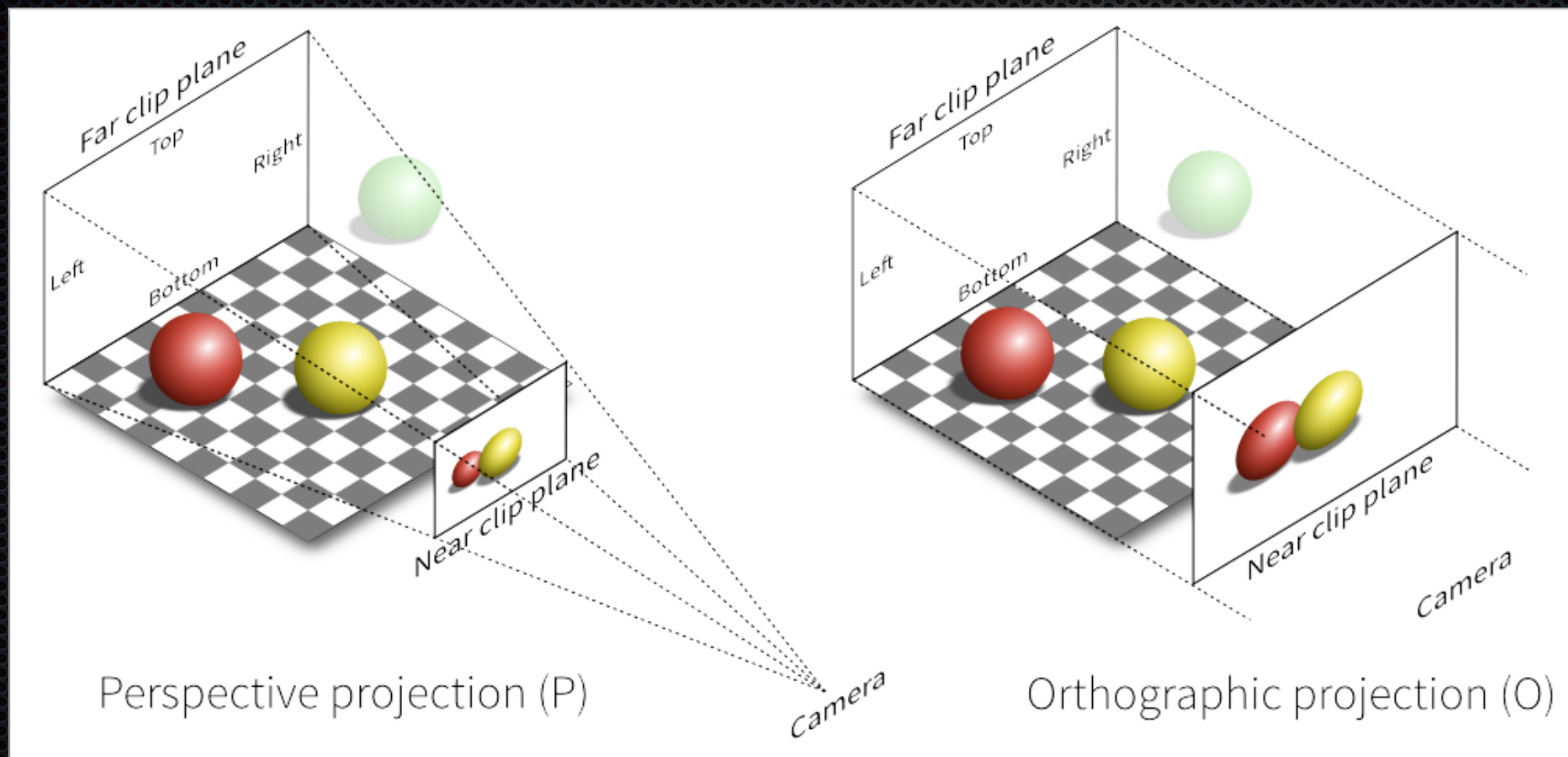
Perspective Projection

- ✦ normal projection with
 - ✦ field of view
 - ✦ aspect ratio
 - ✦ near and far clipping plane

Orthographic Projection

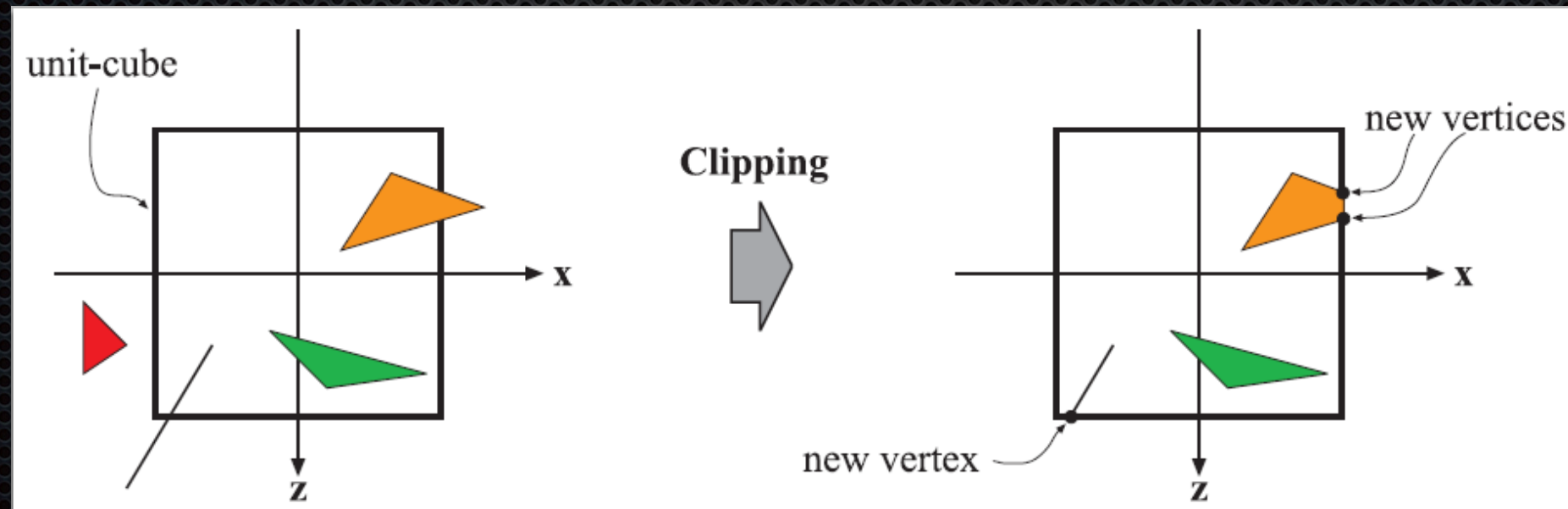
- ✦ perpendicular projection with
 - ✦ window size
 - ✦ near and far clipping plane

Perspective vs. Orthographic Projection



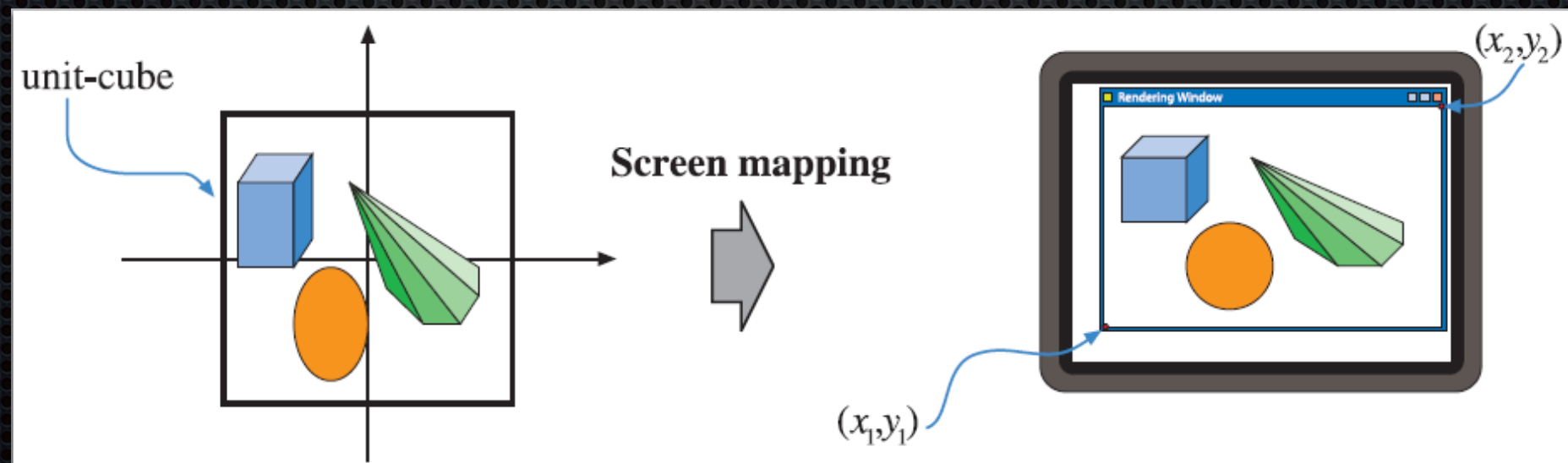
Clipping

- ✦ objects outside unit cube will be discarded
- ✦ objects partly outside will be recreated



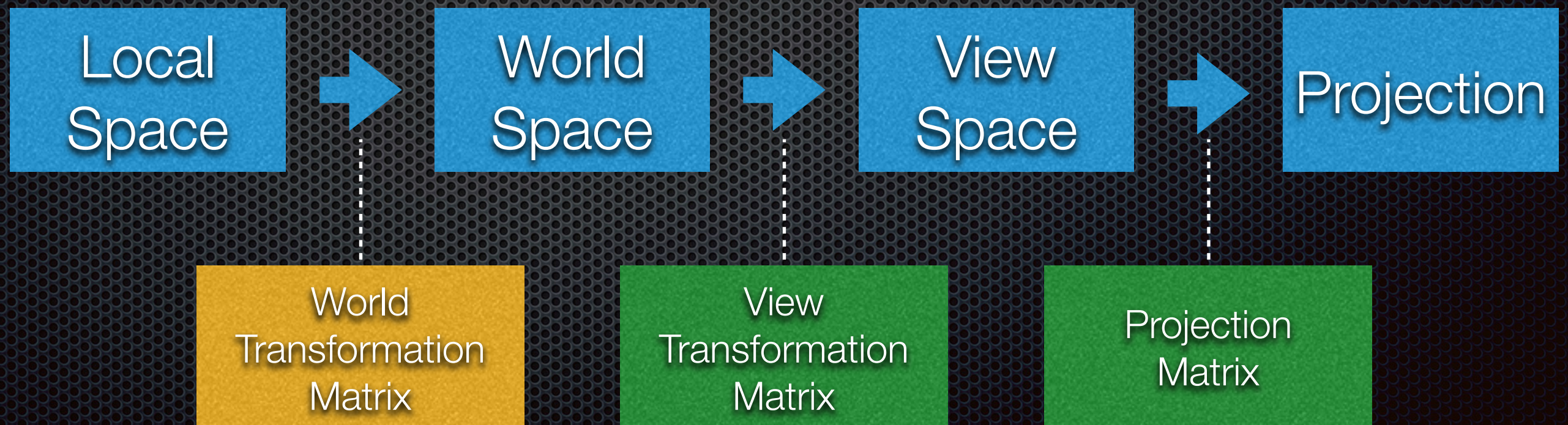
Screen Mapping

- ✦ transform from normalized to screen space coordinates
- ✦ along z-axis
- ✦ $[+0, +1]$



Transformation

- every object has to pass some stages
- every stage consists of a transformation
- for calculation we use matrices



DirectX & Mathematics

- DirectXMath
 - new library
 - SIMD friendly
 - difference between data storage and calculation
 - XMFLOAT... for storage
 - XMVECTOR / XMATRIX for calculation
 - DirectX Tool Kit (only 11) - SimpleMath is wrapper

Coding Time

Algorithm (noun.)

Word used by programmers when...
they do not want to explain what they did.

Input Handling

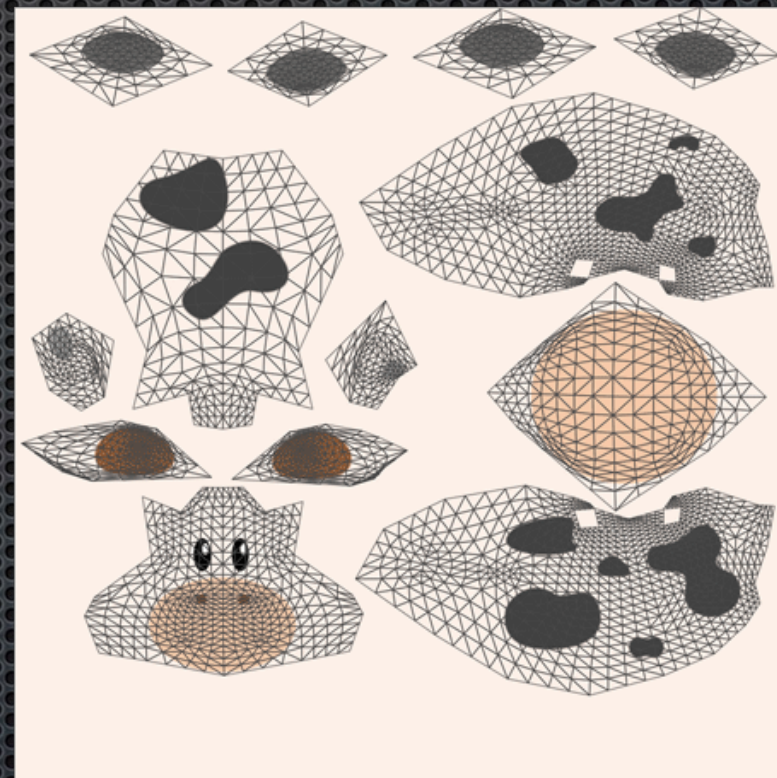
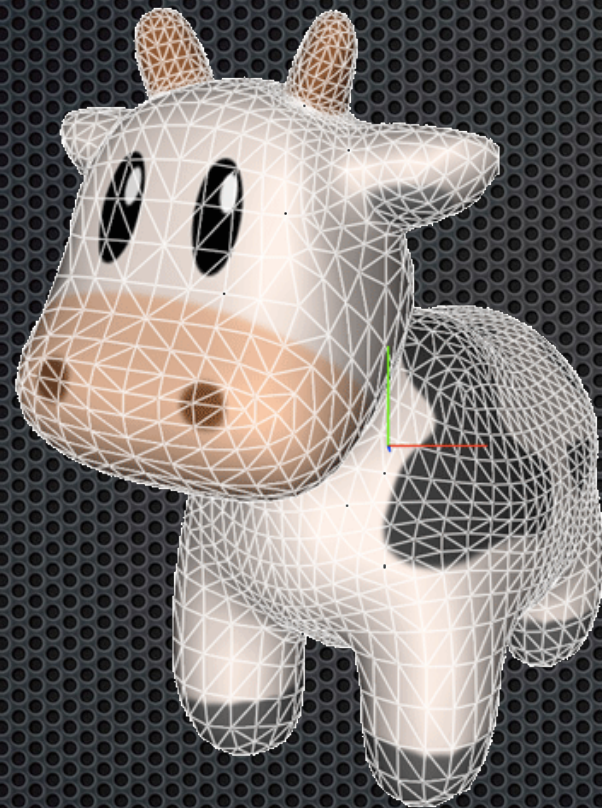
- Window messages
 - WM_KEYDOWN / WM_KEYUP
 - WM_LBUTTONDOWN / WM_LBUTTONUP / WM_MOUSEMOVE
 - WM_INPUT
- Get(Async)KeyState / GetKeyboardState
- DirectInput
- XInput

Time

- ✦ <ctime>
- ✦ <chrono>
- ✦ GetTickCount
- ✦ timeGetTime
- ✦ High Performance Timer

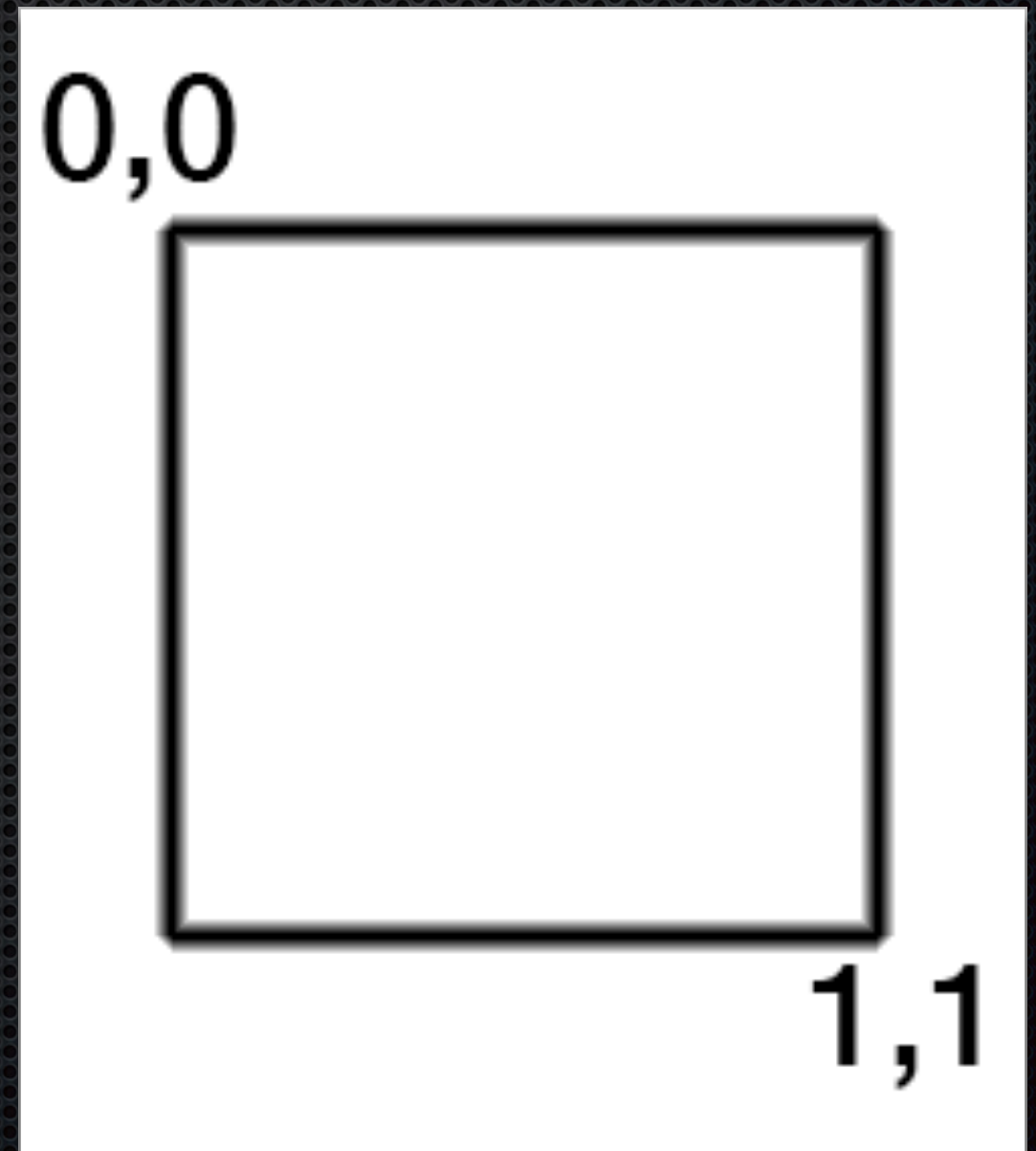
Texture Mapping

- needs for „skinning“ a mesh



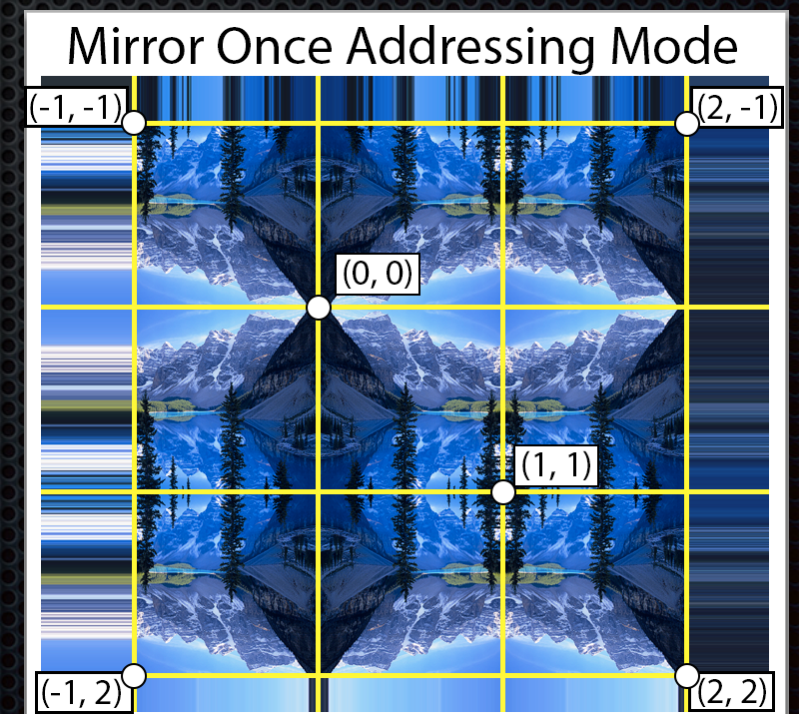
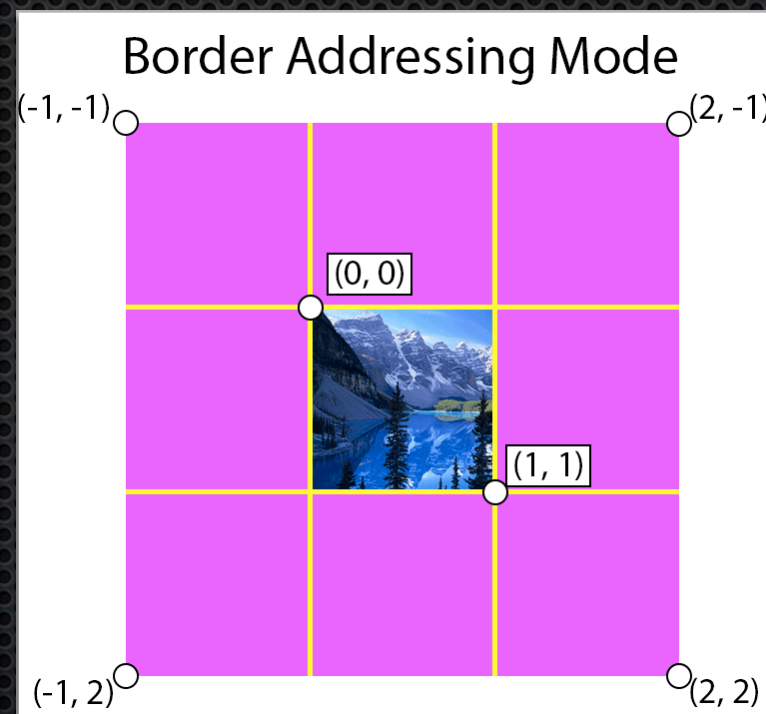
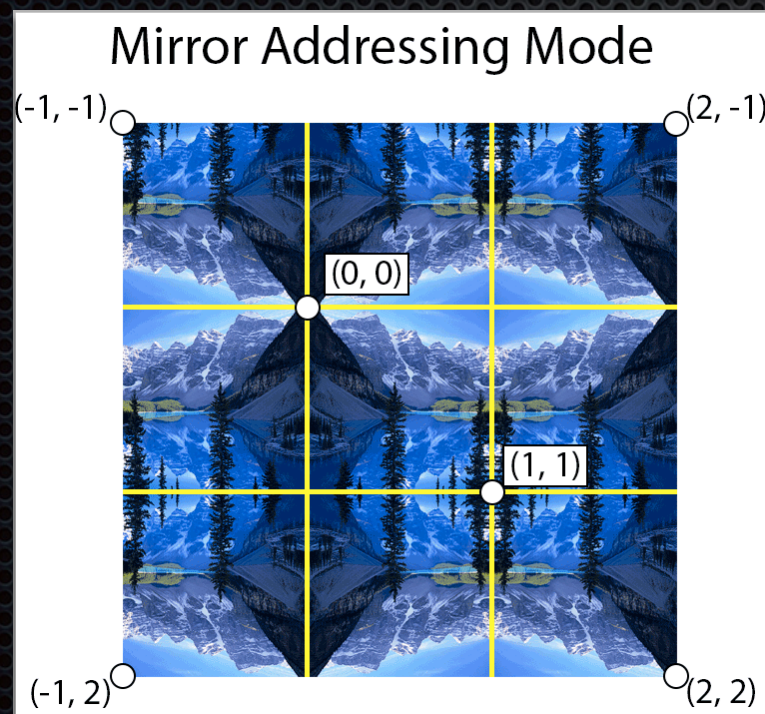
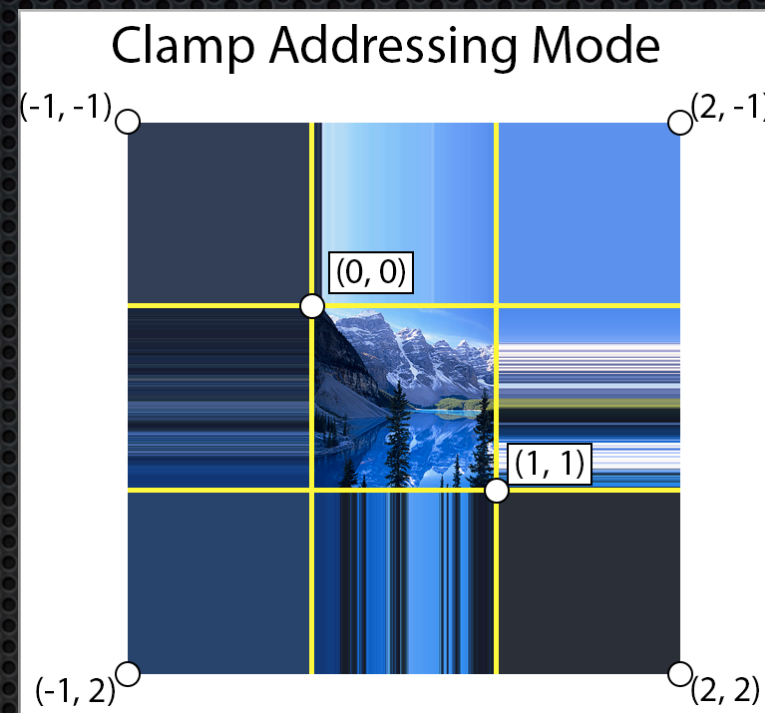
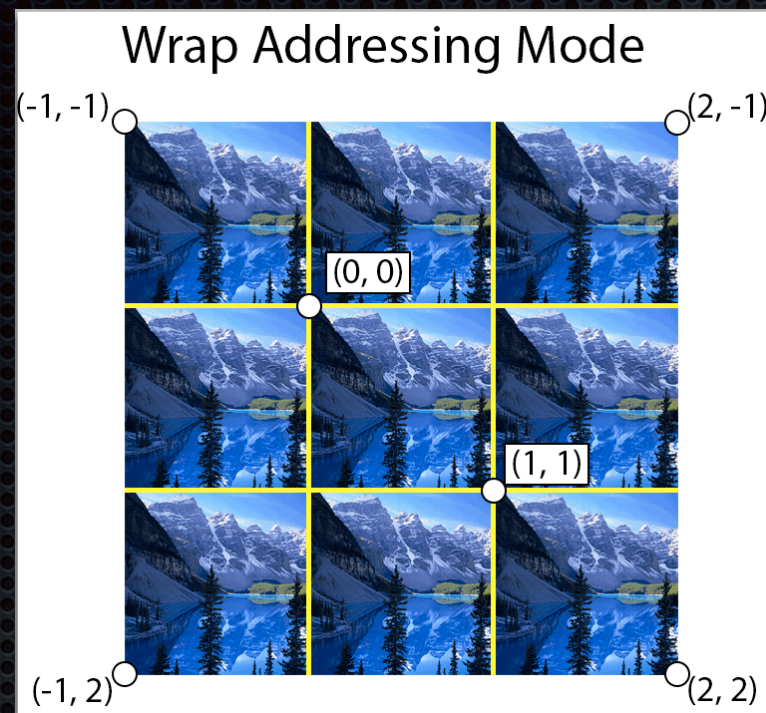
UV Mapping

- ✦ normalized coordinate system
- ✦ original in left top corner
- ✦ every vertex has its own pair of uv coordinates



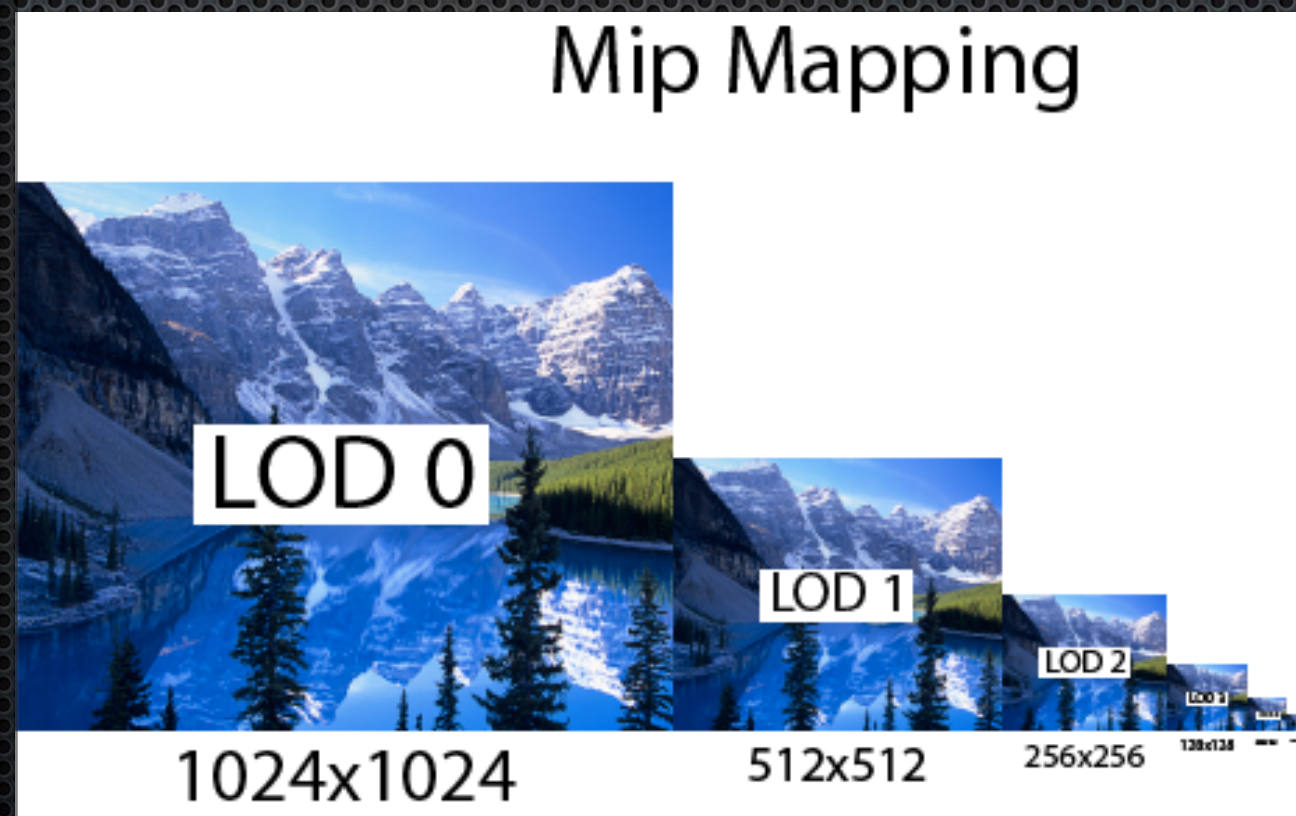
Sampler State I

- a sampler state consists of many settings for sampling a pixel on a texture referenced by uv coordinates
- address mode set the behaviour of mapping outside the normalised coordinate system
- address modes are
 - WRAP
 - CLAMP
 - MIRROR
 - BORDER
 - MIRROR ONCE



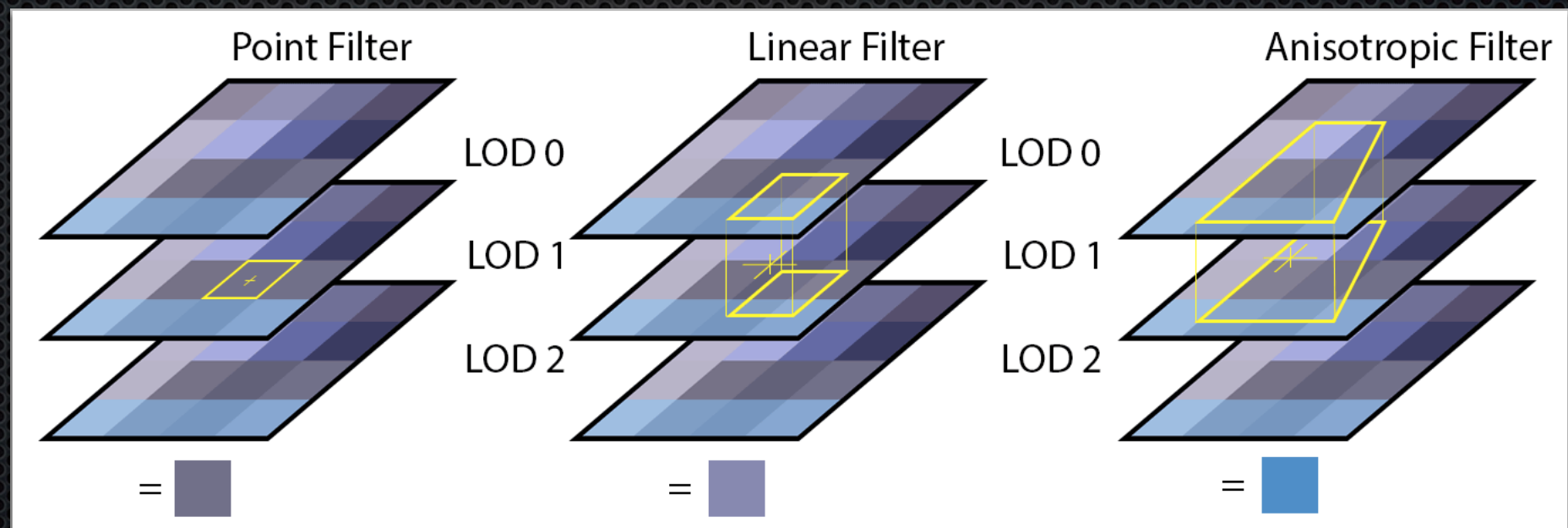
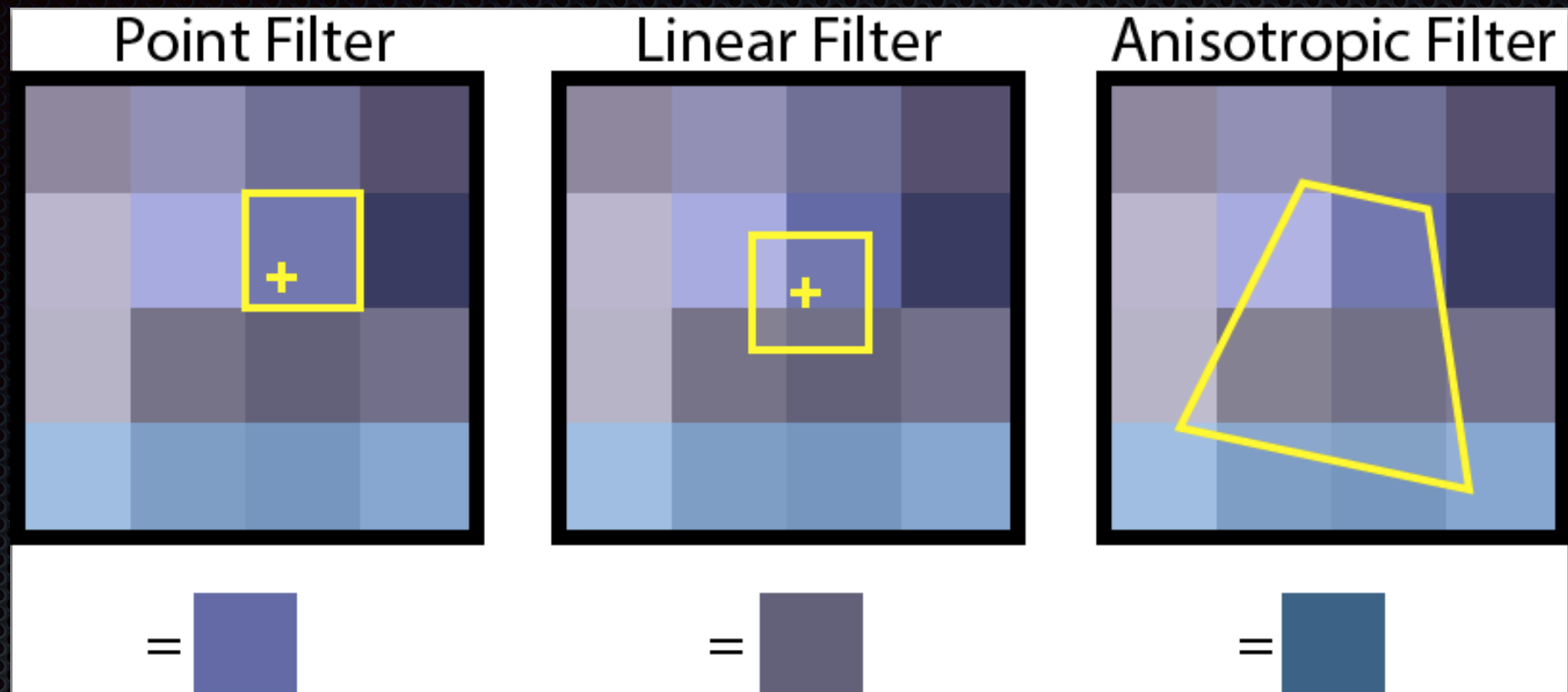
Sampler State II

- ✦ Mip Mapping creates smaller variants of a texture
- ✦ needs for better performance and caching issues



Sampler State II

- ✦ a filter distinguish the sampled color of pixel
- ✦ modes are
 - ✦ point filtering
 - ✦ linear filtering
 - ✦ anisotropic filtering



Point

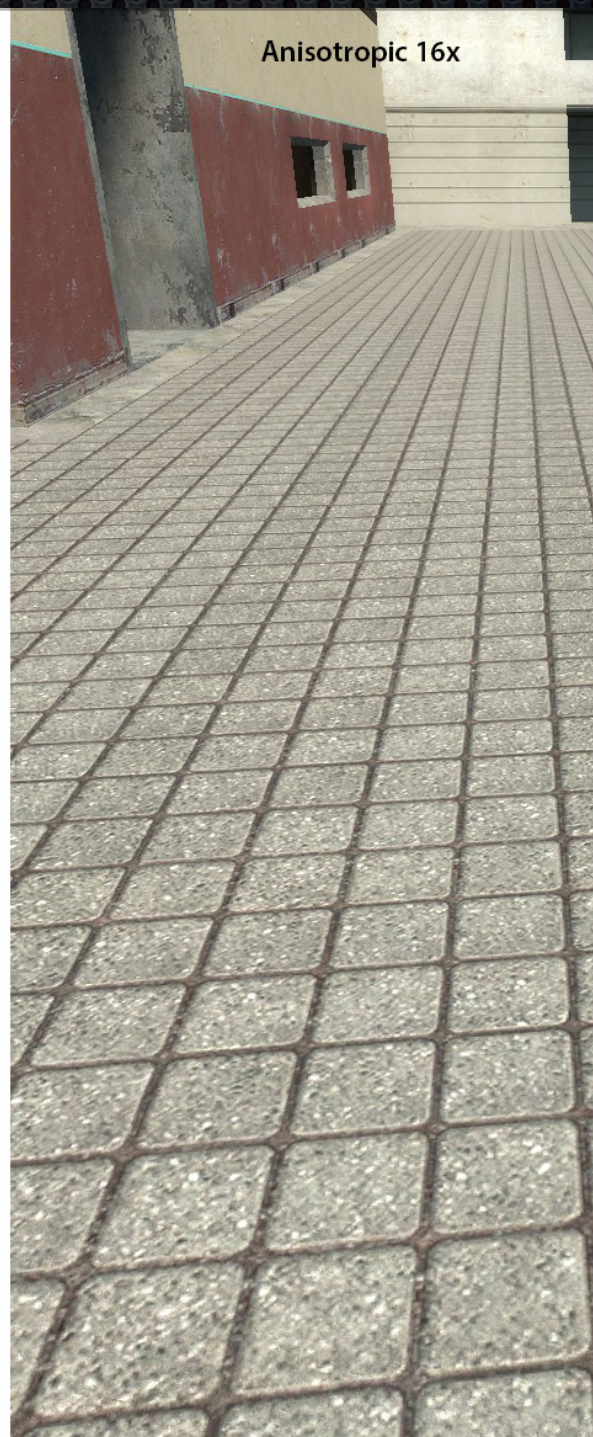
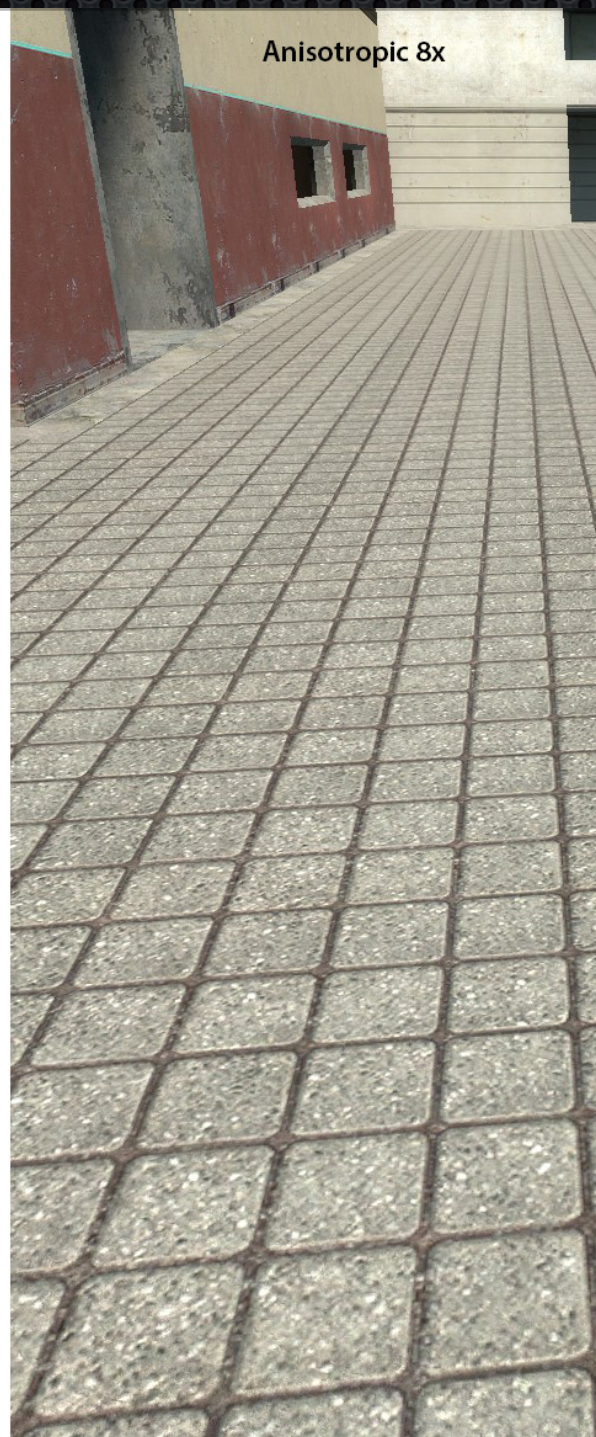
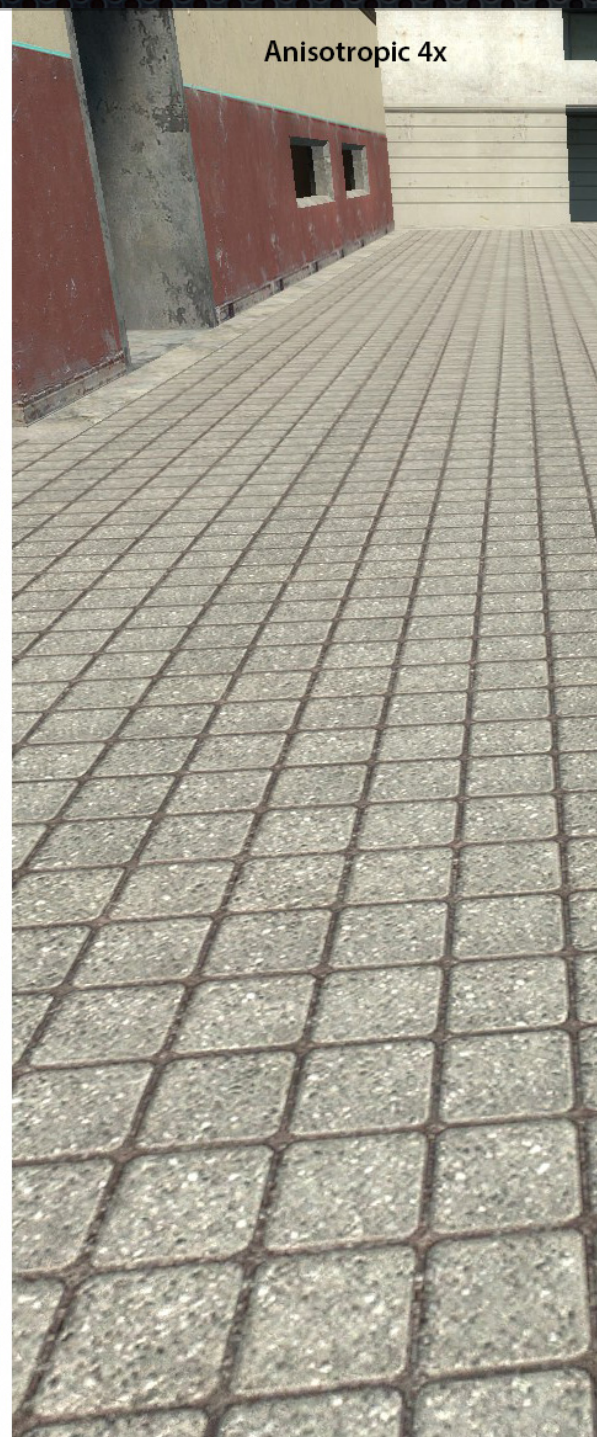
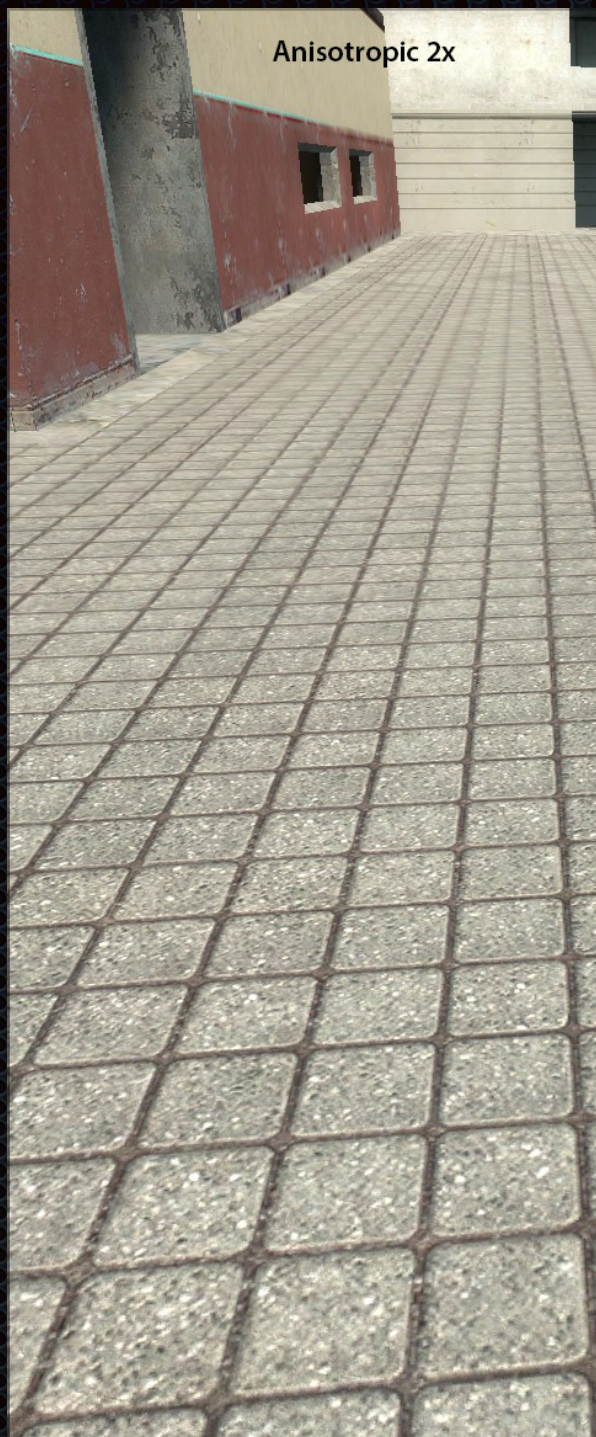


Linear



Anisotropic





Coding Time

Programmer (noun.)

A person who fixed a problem that
you don't know you have,
in a way you don't understand.

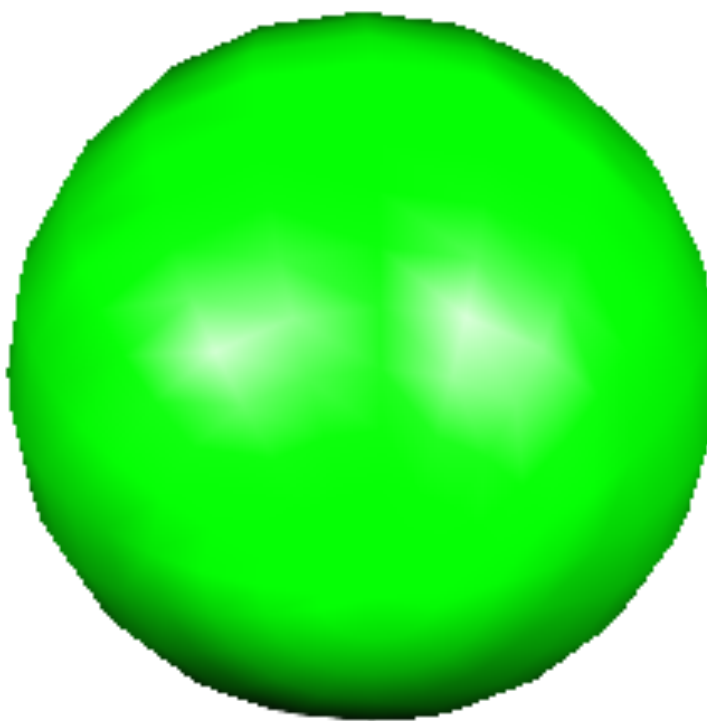
Lighting

- ✦ lighting per vertex -> Vertex Lighting
- ✦ not usual today
- ✦ not so precise
- ✦ better: Pixel Lighting
- ✦ at least save information for later lighting

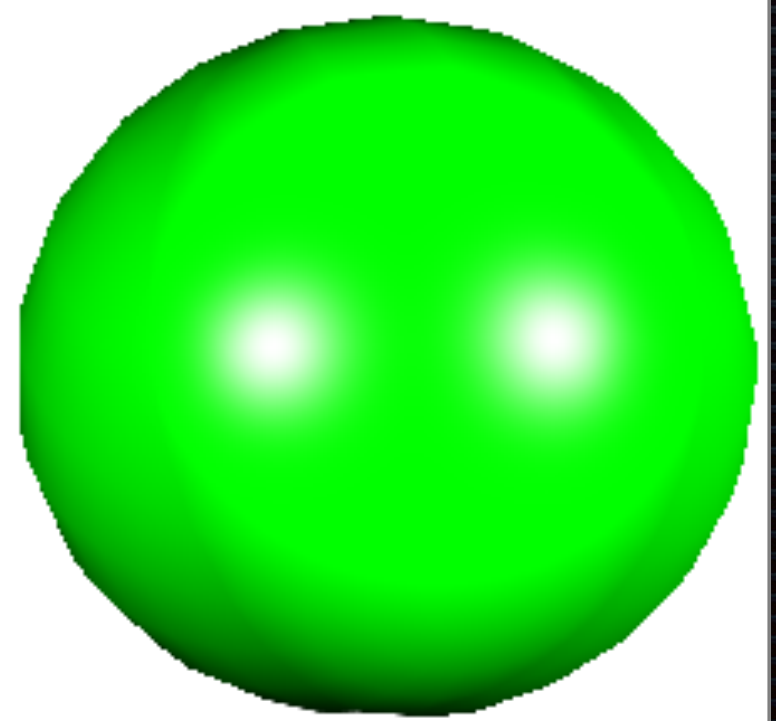
Face vs. Vertex vs. Pixel Lighting



Per-face



Per-vertex



Per-pixel

Sources

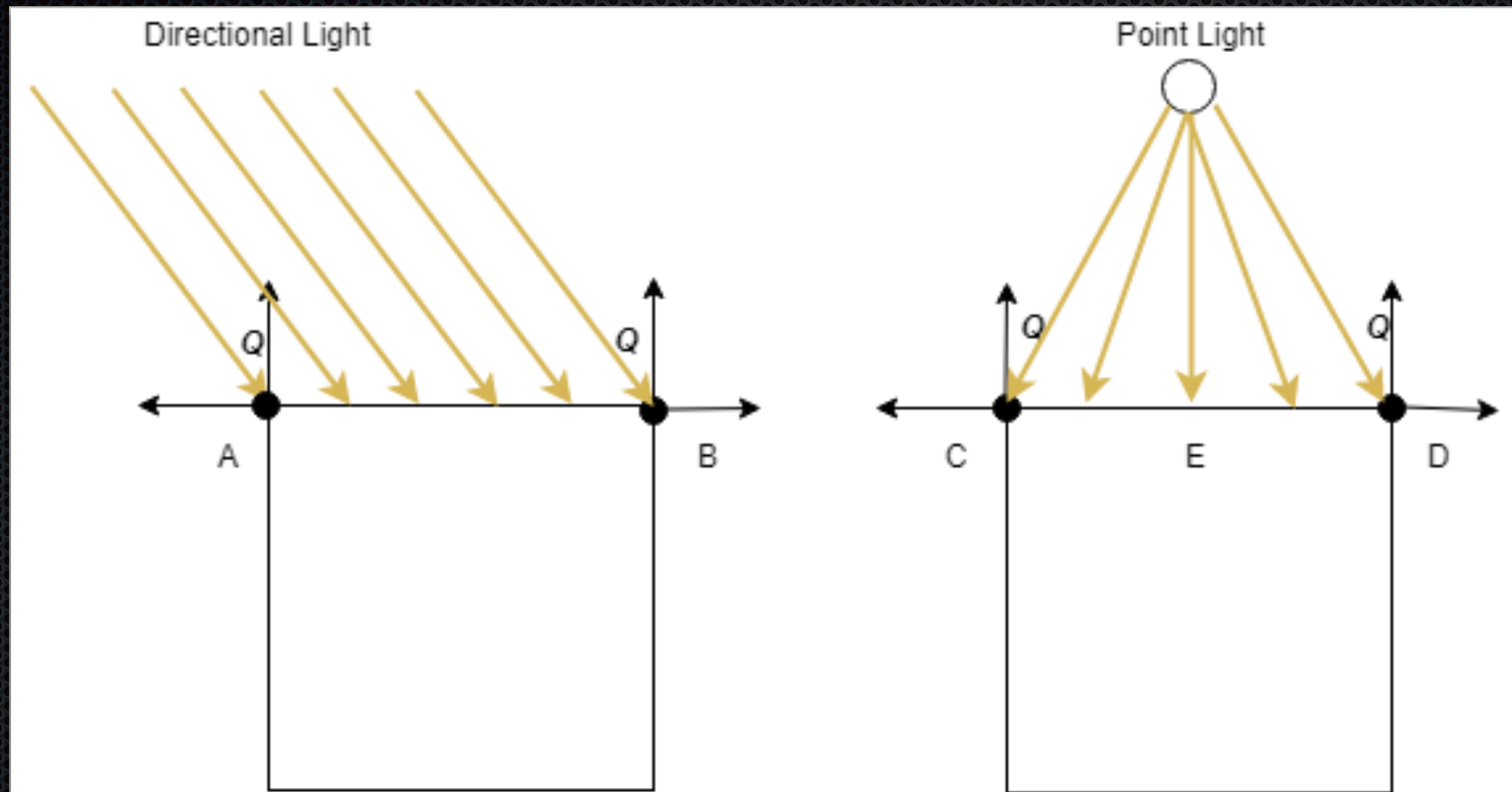
- ✦ Directional Light
- ✦ Point Light
- ✦ Spot Light

Types

- ✦ Ambient
- ✦ Diffuse
- ✦ Specular
- ✦ Emission

$\text{Texture} * (\text{Ambient} + \text{Diffuse}) + \text{Specular} + \text{Emission}$

Light Source



Light Source



L



N



R



Viewer



V



Surface



$$n \cdot l = \|n\| \|l\| \cos \theta$$

Coding Time

Have a great weekend
I hope your code behaves on Monday
the same way it did on Friday



STARECAT.COM