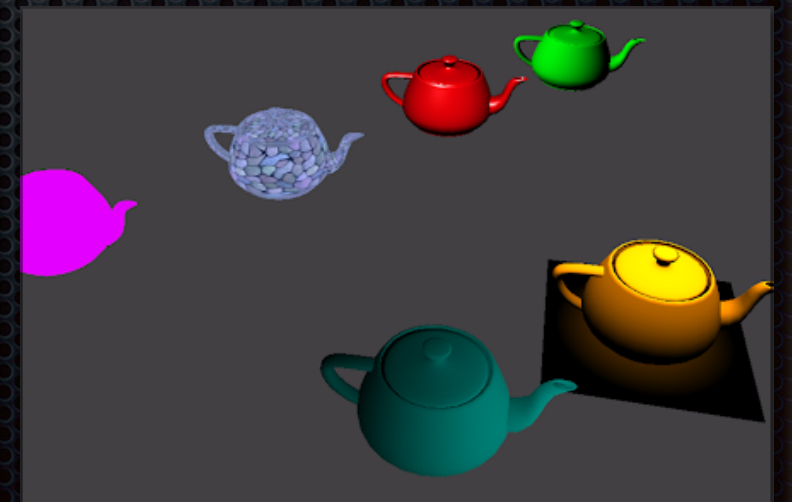
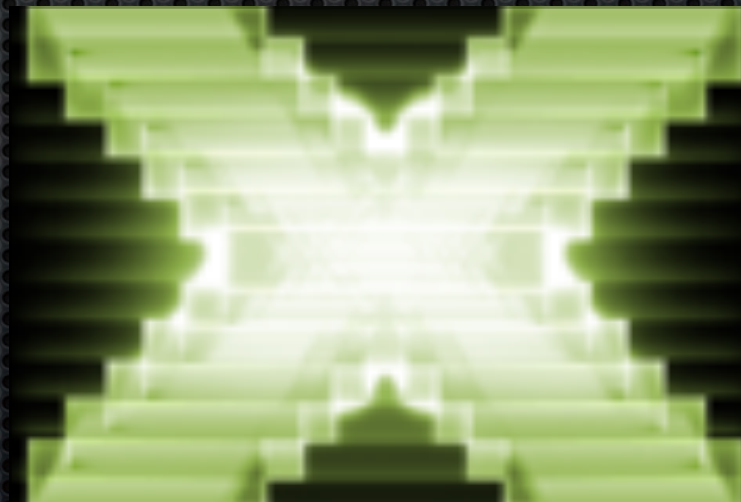


Graphical Data Processing

David Hackbarth

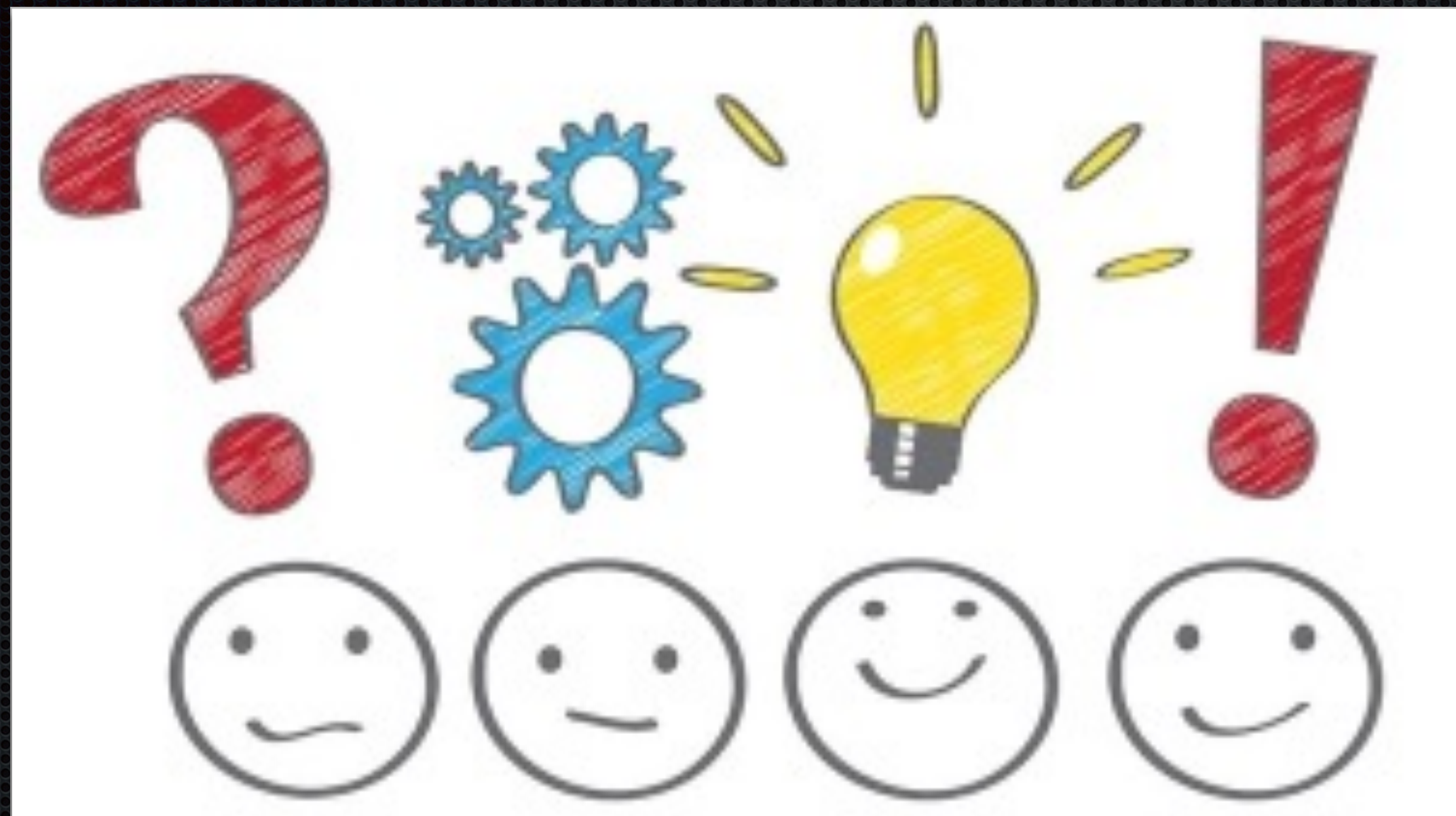


Schedule

- ✦ Theory
- ✦ Window
- ✦ Fixed-Function Pipeline
- ✦ Programmable Pipeline
- ✦ (Unity Shader)

Exam

- ✦ 2 Hours
- ✦ Written
- ✦ Knowledge (50%)
- ✦ Use of Knowledge (40%)
- ✦ Combined Knowledge (10%)
- ✦ No Programming



History

Vector graphics

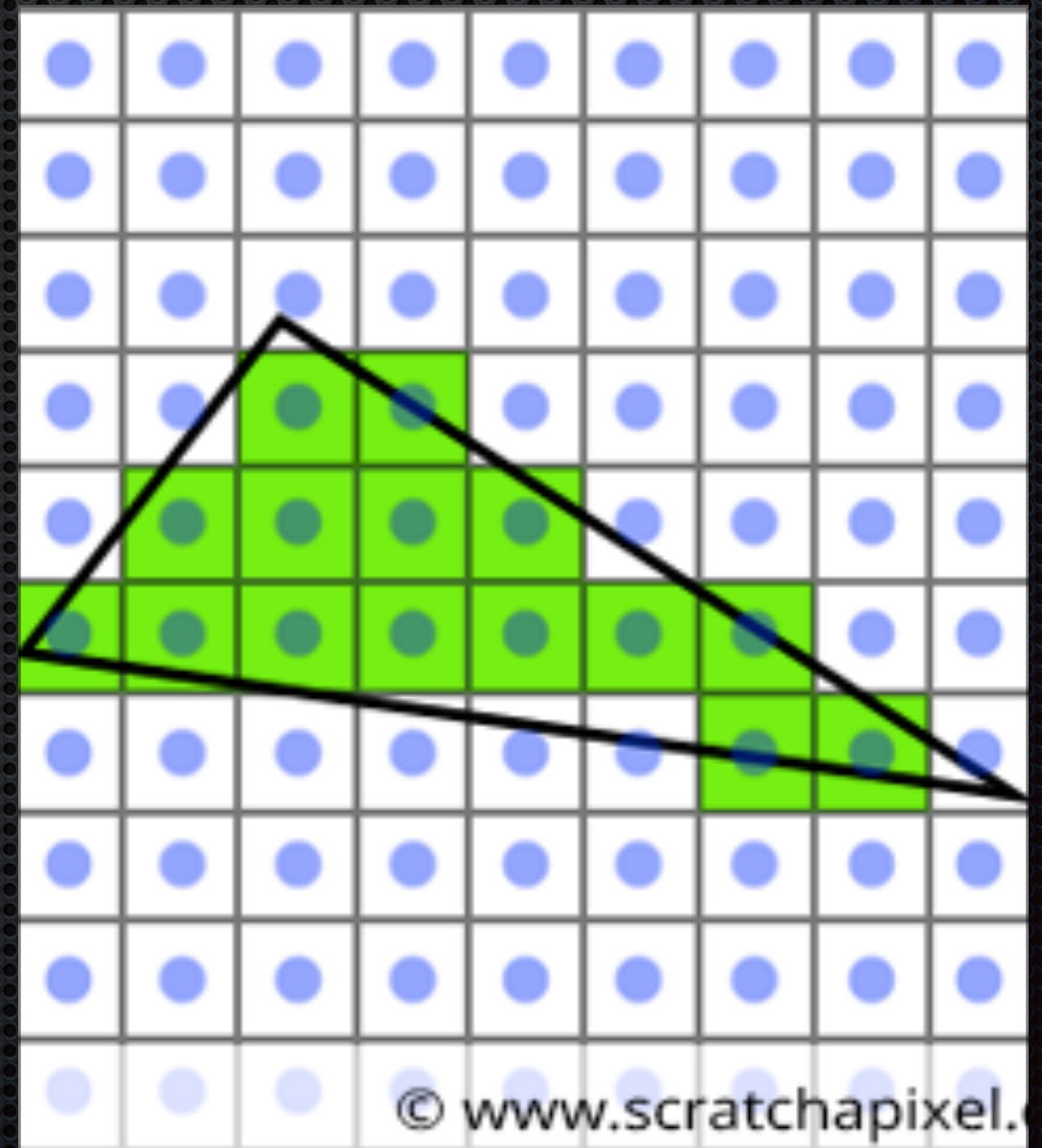


- ✦ 1958 - Tennis For Two (William Higinbotham)
- ✦ 1963 - Sketchpad (Ivan Sutherland)
- ✦ inflexible



Raster graphics

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



Hardware Sprites

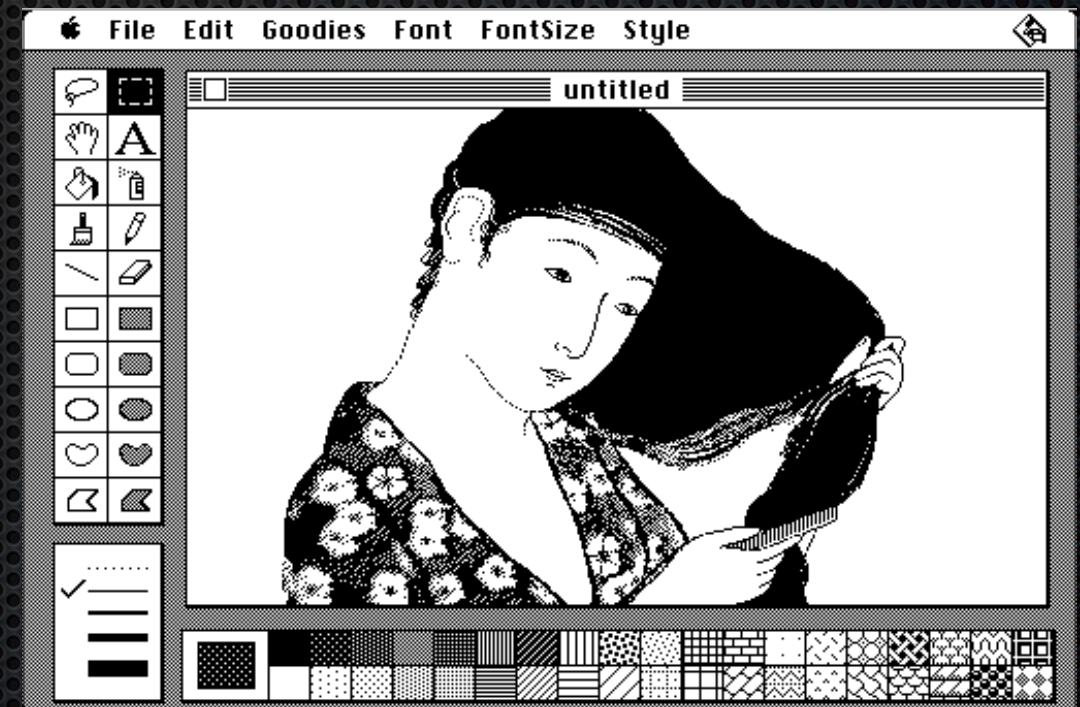


- ✦ 1970s
- ✦ limited size
- ✦ limited colors
- ✦ limited amount on screen



Graphical User Interface

- ✧ Mid 1980s
- ✧ GUI
- ✧ Games



1990s

- ✦ Hardware Boost
- ✦ Software Boost
- ✦ 3D Games



Hardware

- ✦ CPU (Pentium)
 - ✦ 1989 - 16 MHz
 - ✦ 1999 - 1400 MHz
- ✦ Graphics Card
 - ✦ 1990 - without acceleration
 - ✦ 1996 - dedicated Graphic Cards



Software

- ✦ Graphics API
 - ✦ OpenGL
 - ✦ DirectX
- ✦ Operating System
 - ✦ MS-DOS
 - ✦ Windows 95



Entertainment

- ✦ 2D Games
- ✦ 3D Games
- ✦ Animationfilms



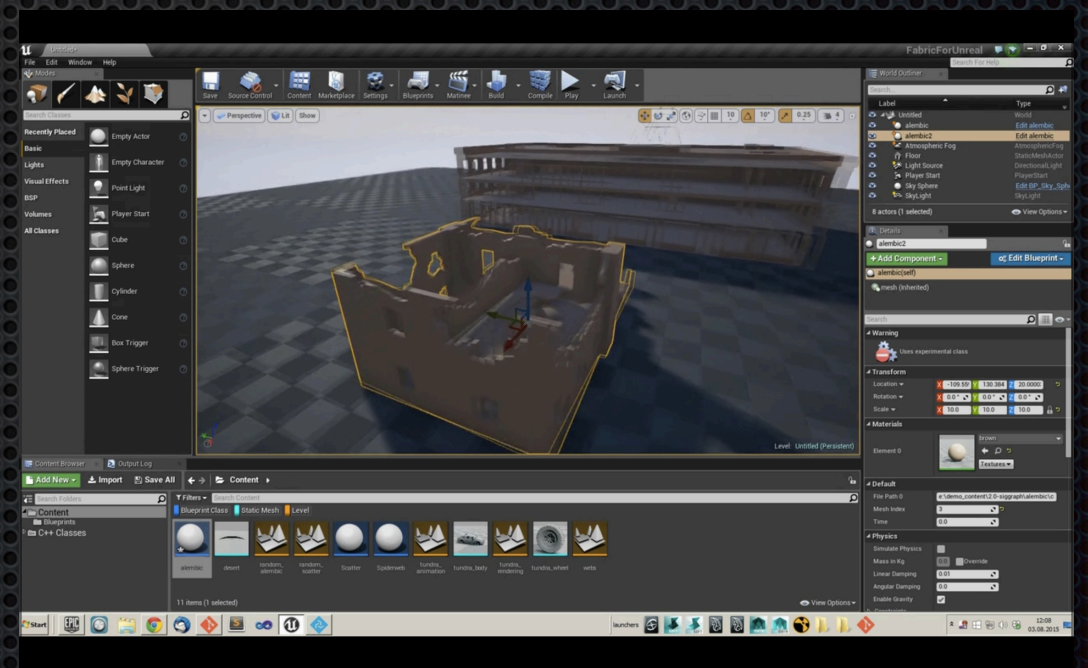
Fixed-Function Pipeline

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



2000s

- ✦ Fast Development
 - ✦ New consoles
 - ✦ CPU/GPU
- ✦ Game Engines



Programmable Pipeline

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



Key Facts

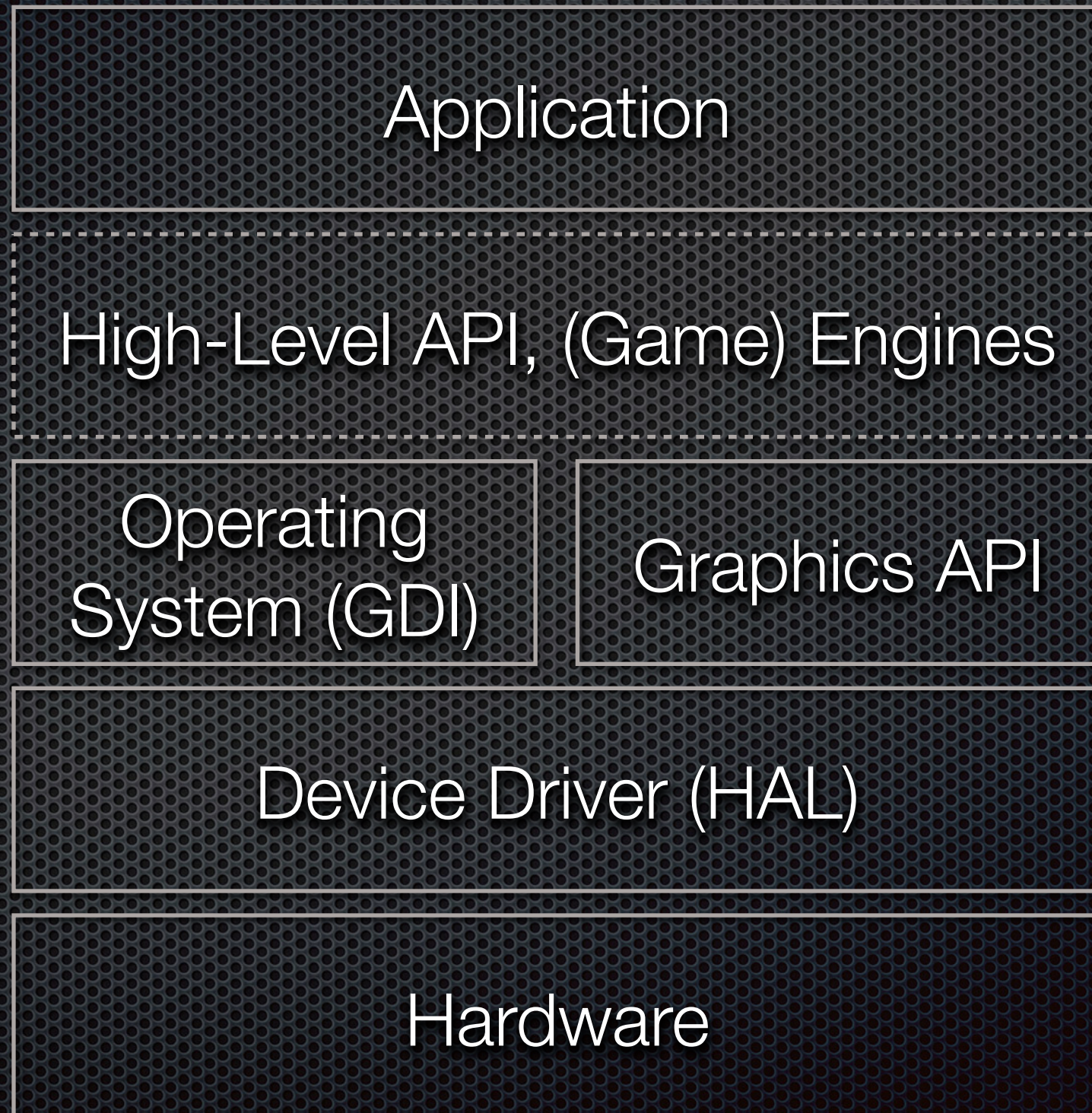
- Increase of CPU and GPU performance
- Better graphic effects
- Faster rendering times in Realtime and for Animation Films
- Motion Capturing
- Voxel-Engines
- Stereoscopy
- WebGL / Vulkan / DirectX 12
- Cloud Gaming

Examples

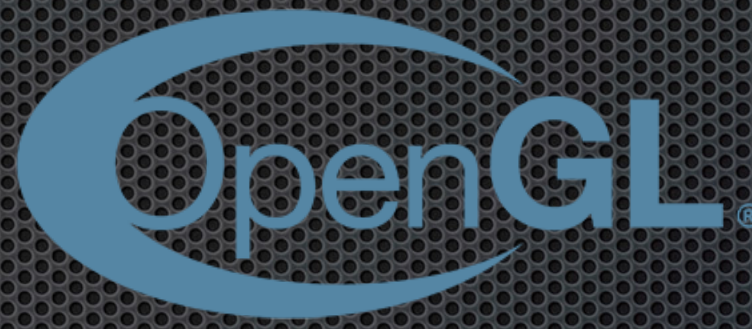
- ✦ Unreal 5.4 Demo
- ✦ Real or Photoshop
- ✦ VFX
- ✦ Film CGI
- ✦ Pixar short films
- ✦ Game Engine Short Film

Graphics API

Abstract Layer



Overview

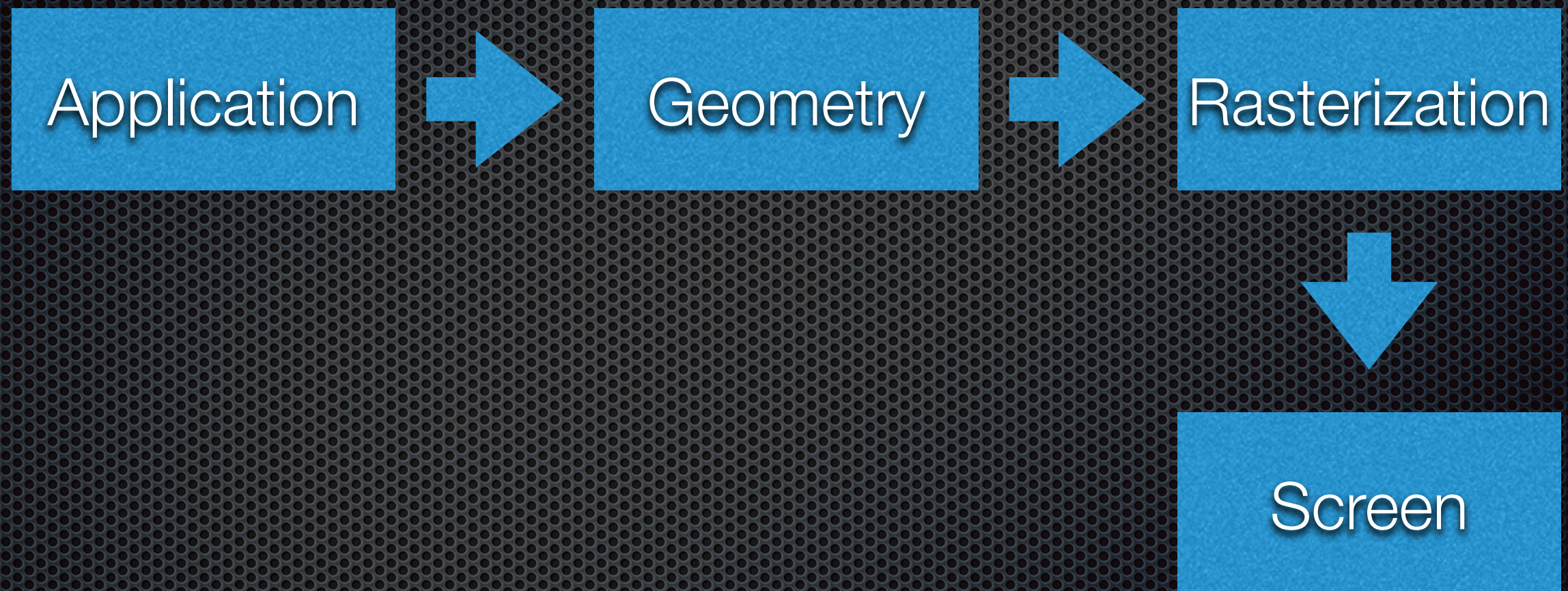


DirectX

- ✦ group of many libraries
 - ✦ Direct3D
 - ✦ DirectSound
 - ✦ DirectInput (new XInput)
 - ✦ Direct2D (removed)
 - ✦ etc.
- ✦ Version: 12
- ✦ too complex for beginners
- ✦ instead 9 and 11

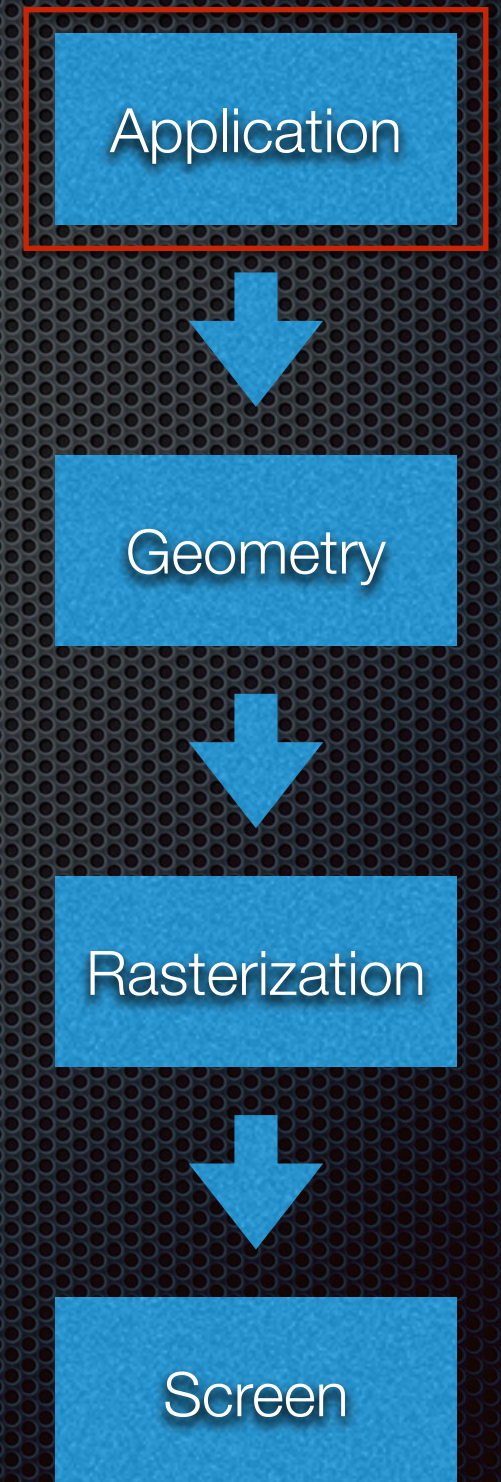
Fixed-Function Pipeline

Structure



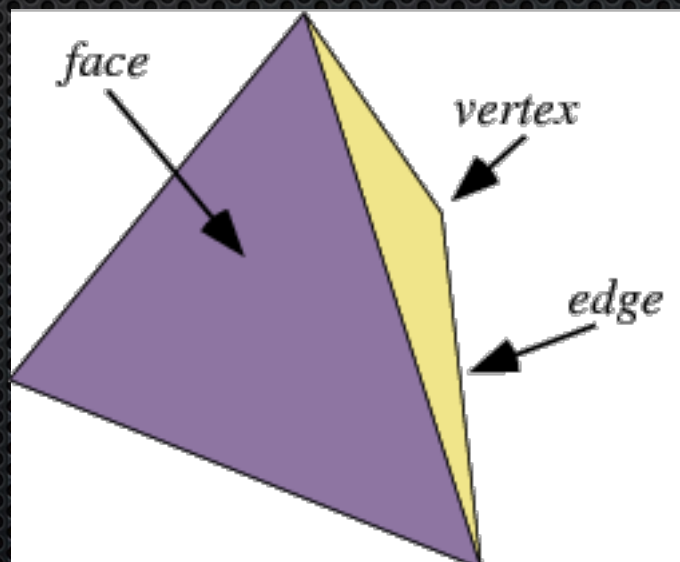
Application

- 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



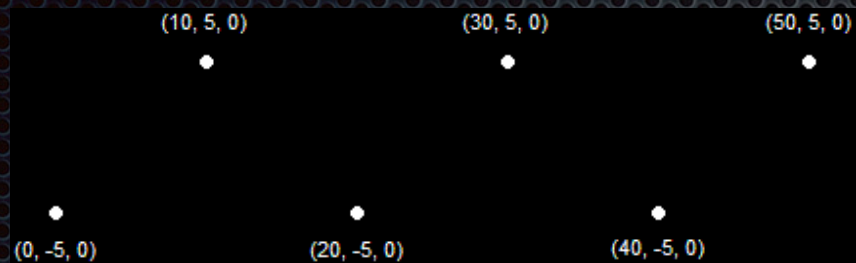
Rasterization



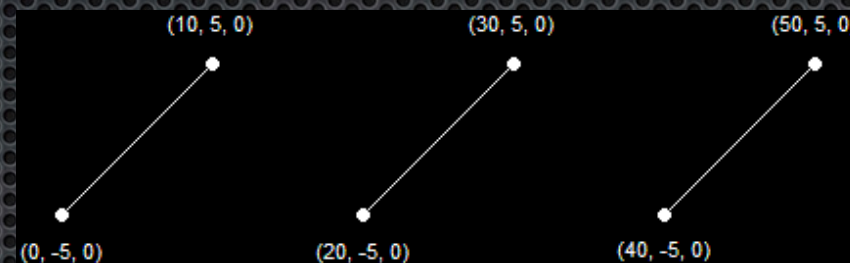
Screen

Primitives

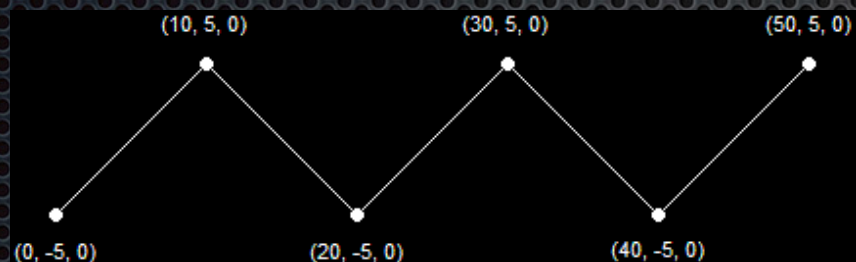
Pointlist



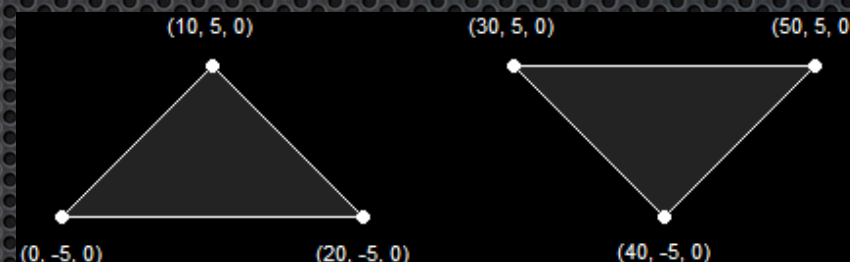
Linelist



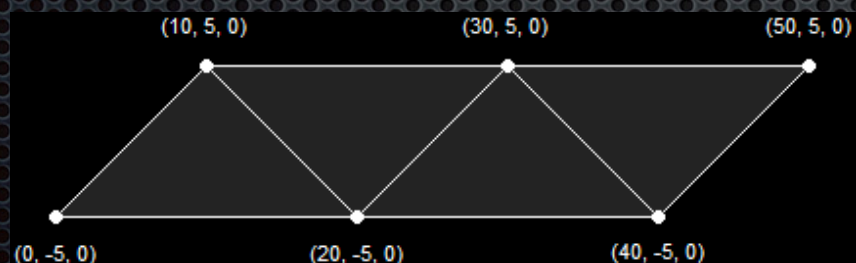
Linestrip



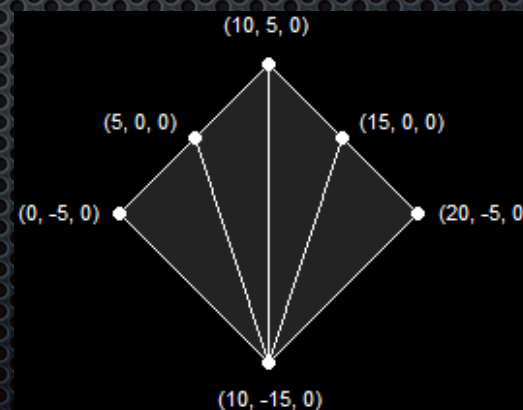
Trianglelist



Trianglestrip



Trianglefan



Application



Geometry



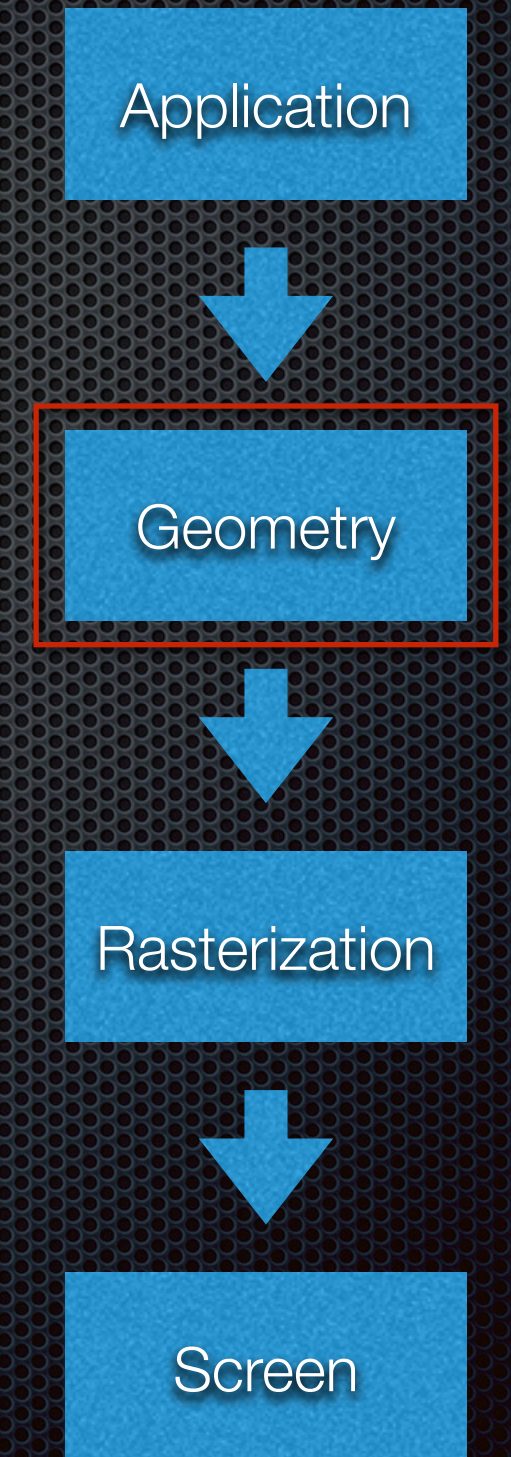
Rasterization



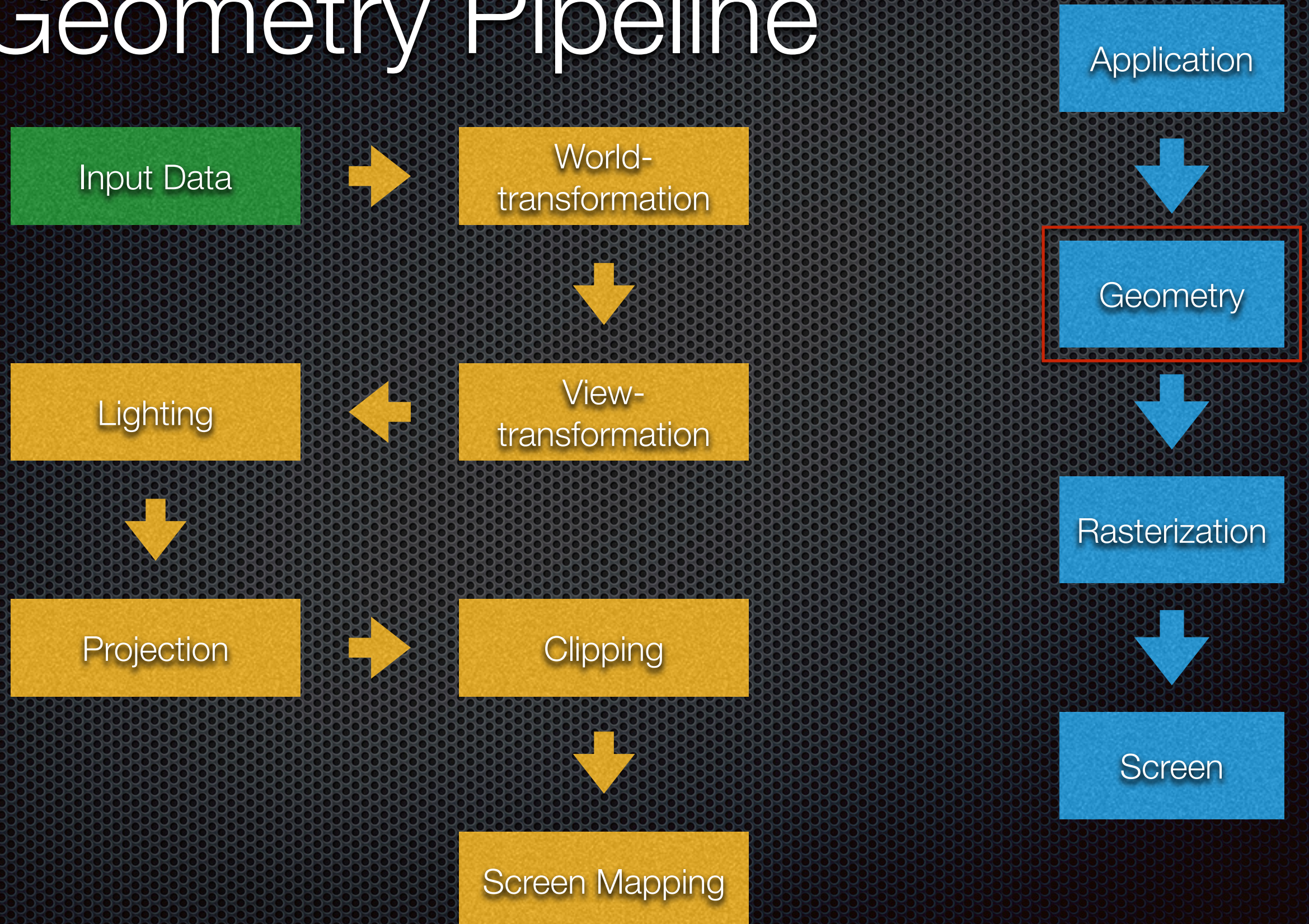
Screen

Geometry

- ✦ transformation
- ✦ lighting
- ✦ projection
- ✦ clipping
- ✦ additional geometry

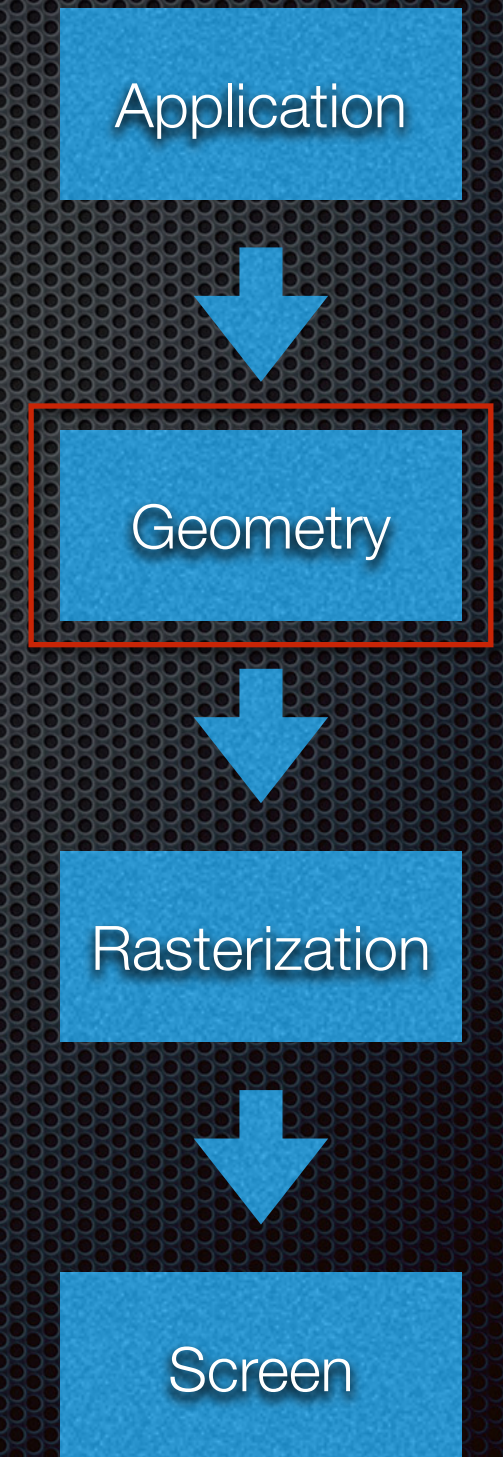


Geometry Pipeline



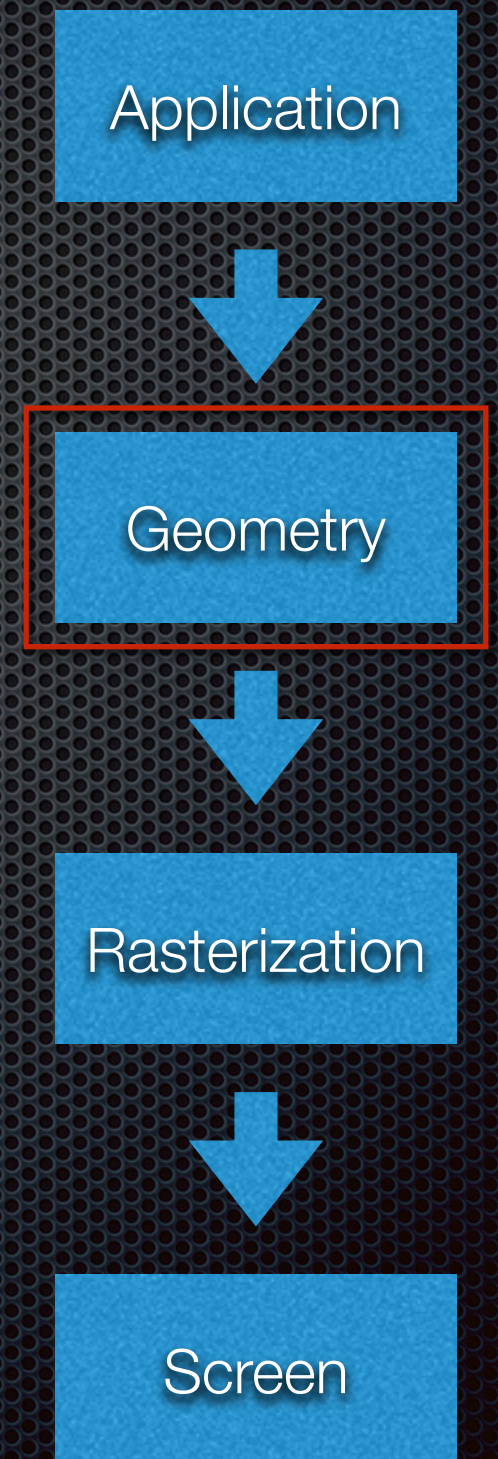
Worldtransformation

- 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

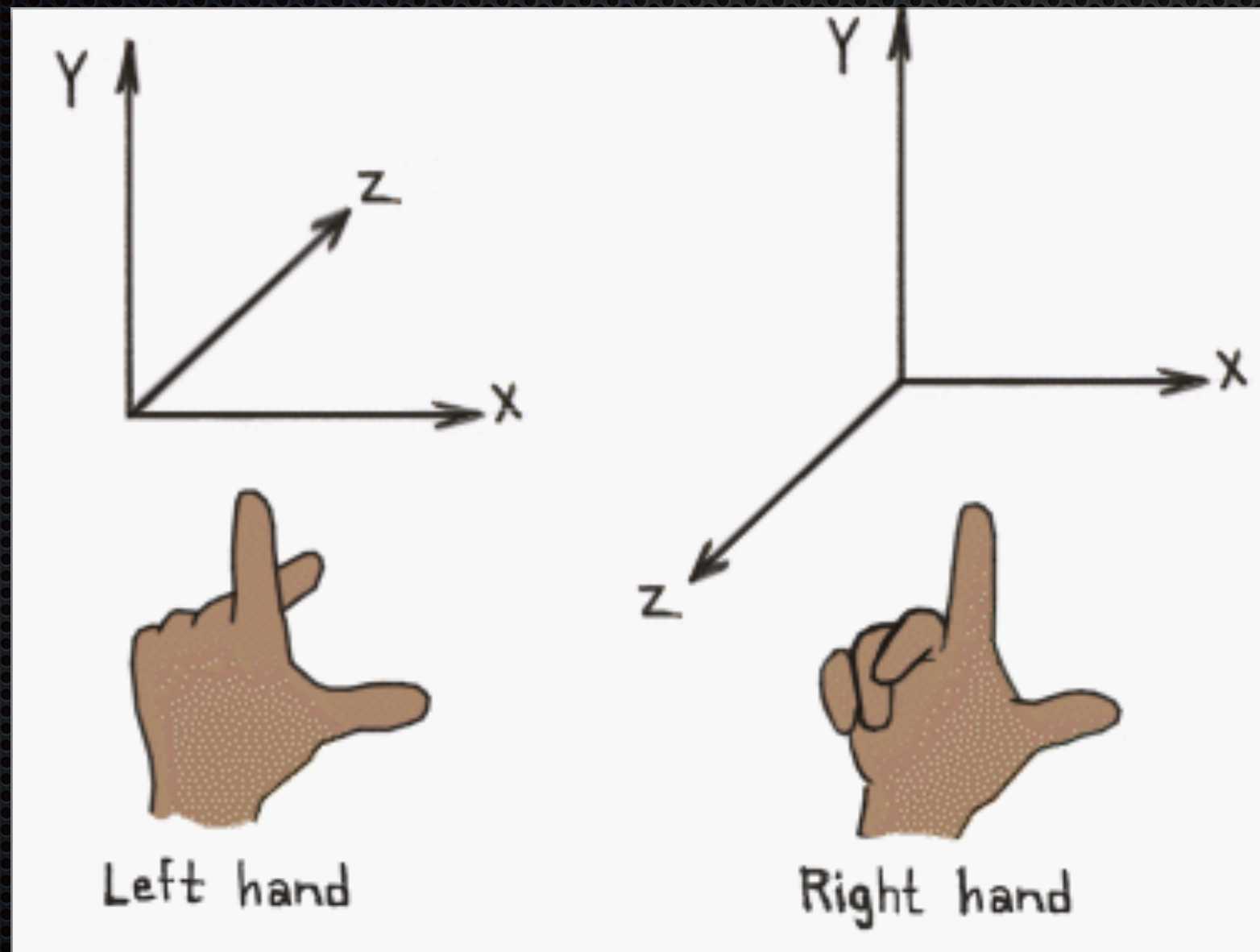


Viewtransformation

- ✦ 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



Application



Geometry

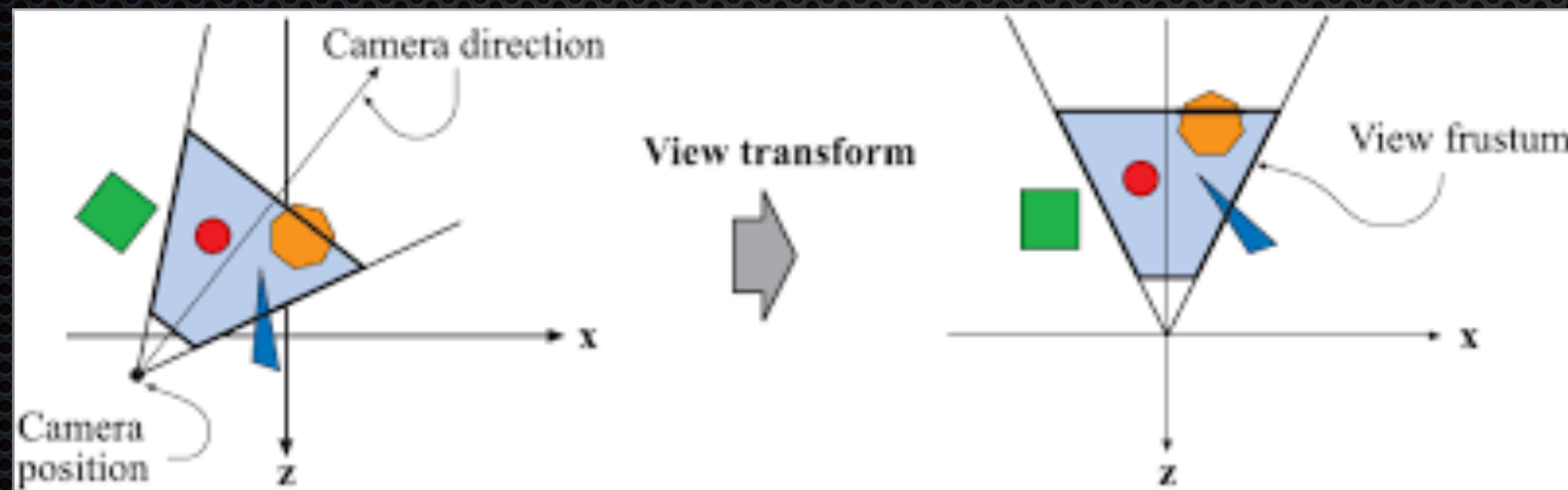


Rasterization



Screen

Viewtransformation



Application



Geometry



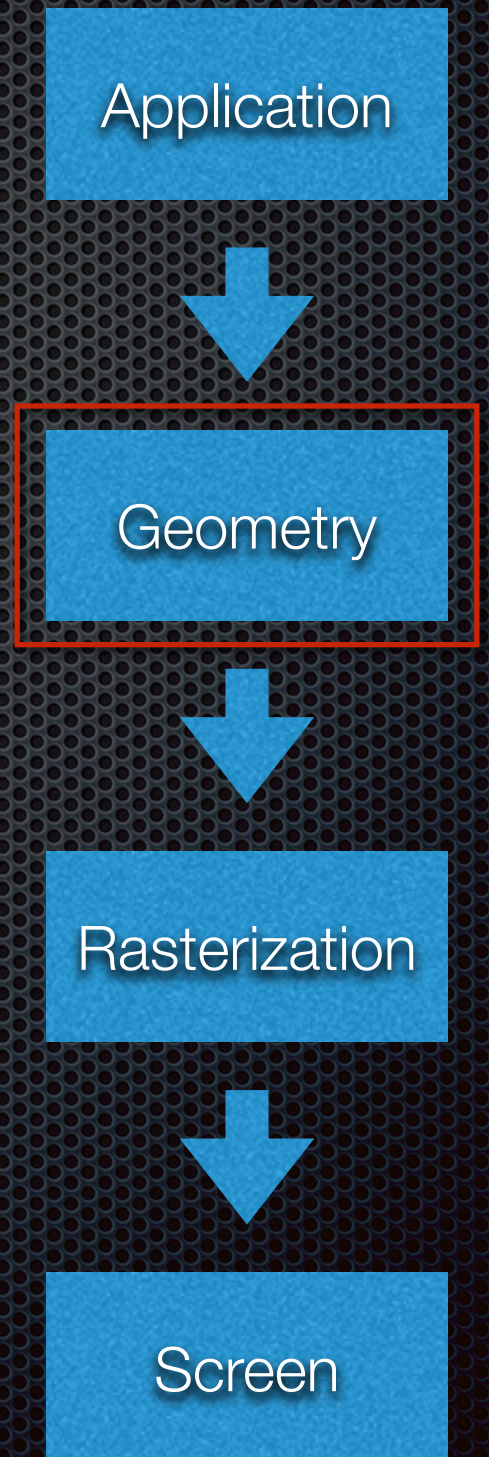
Rasterization



Screen

Lighting

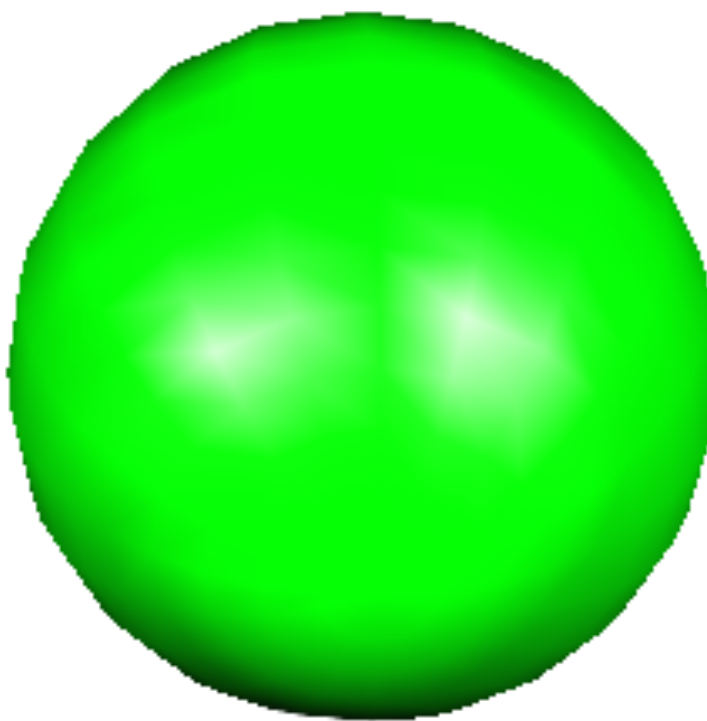
- ✦ lighting per vertex -> Vertex Lighting
- ✦ not usual today
- ✦ not so precise
- ✦ better: Pixel Lighting
- ✦ at least save information for later lighting



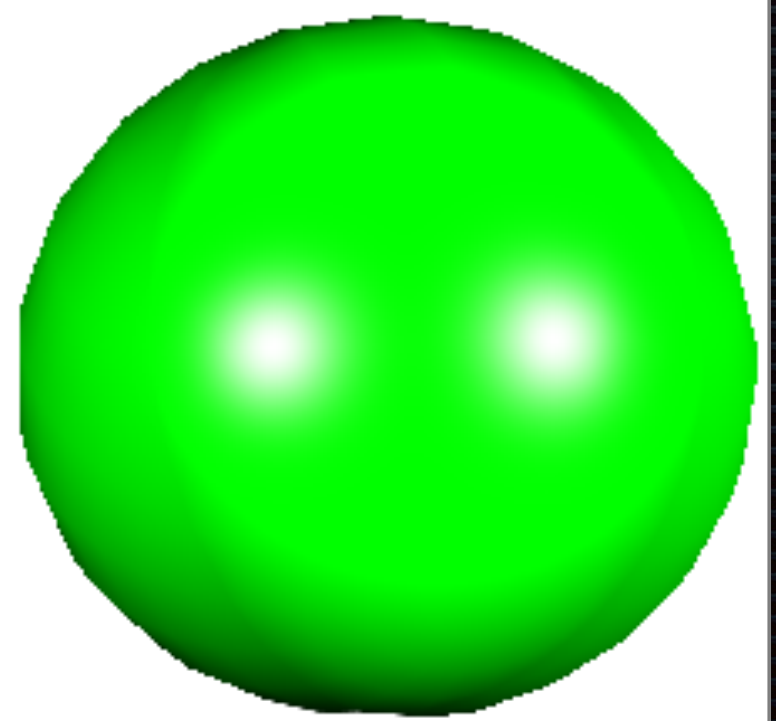
Face vs. Vertex vs. Pixel Lighting



Per-face



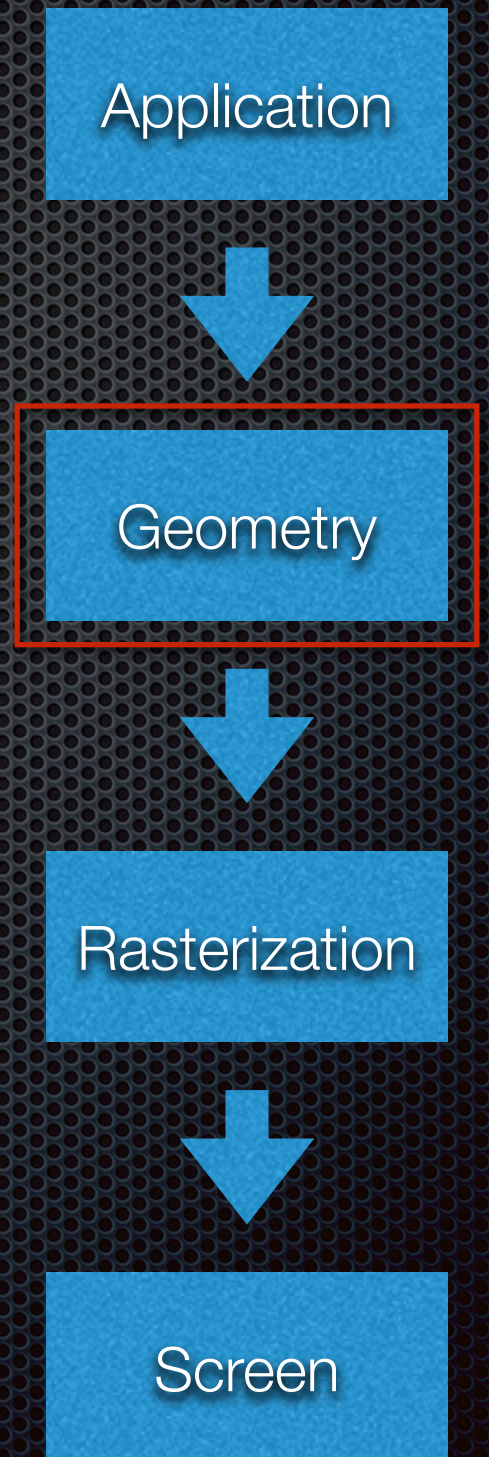
Per-vertex



Per-pixel

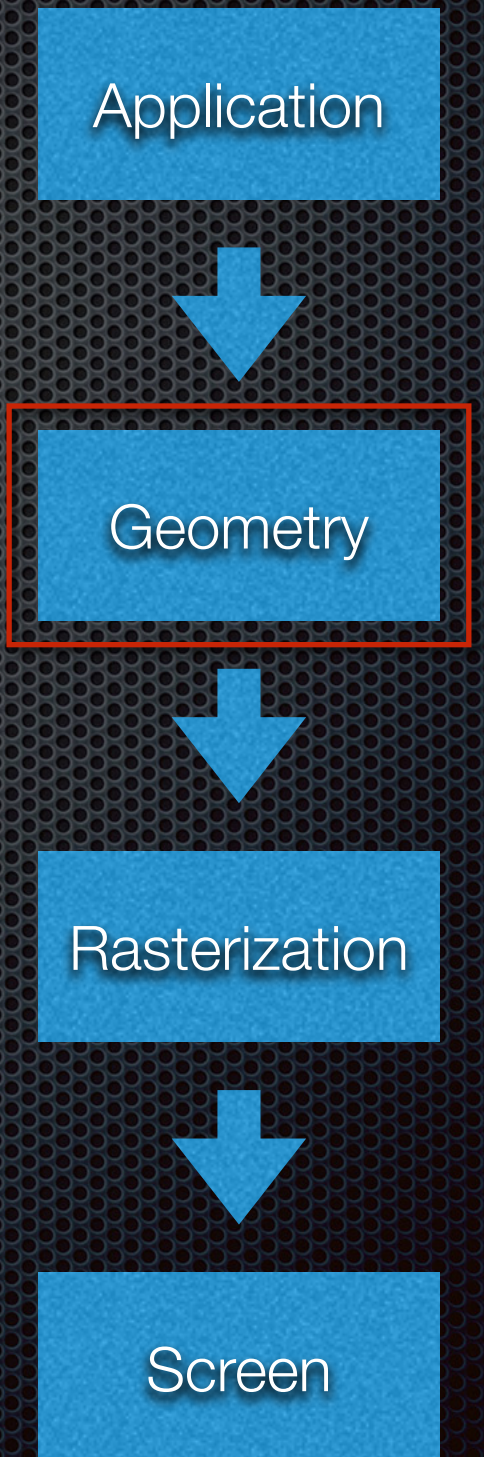
Projection

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



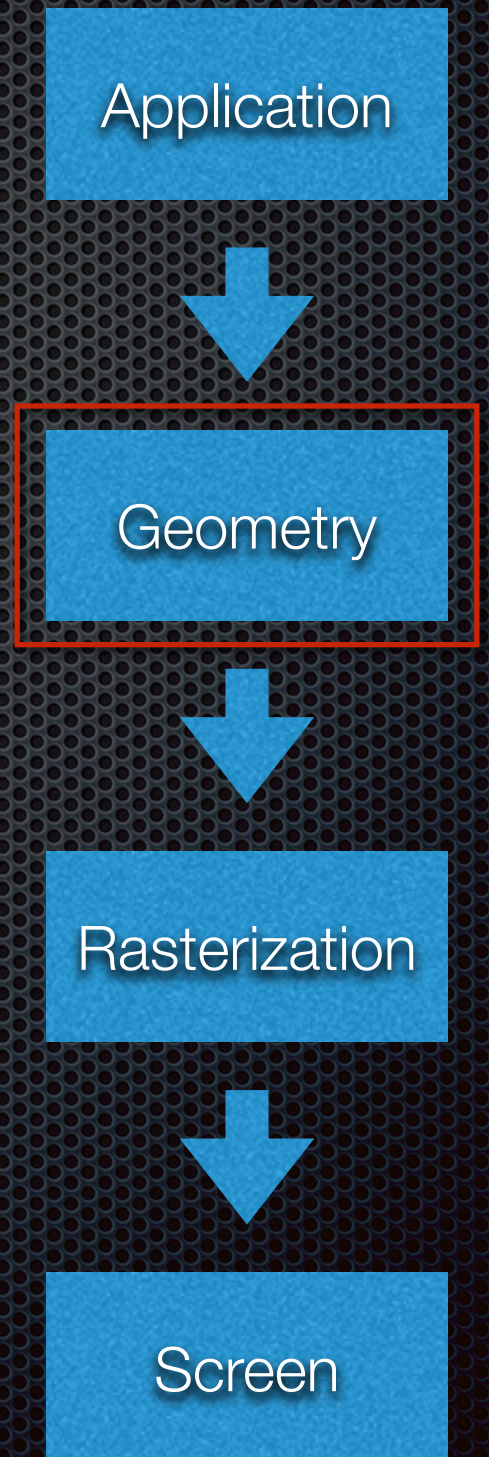
Perspective Projection

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

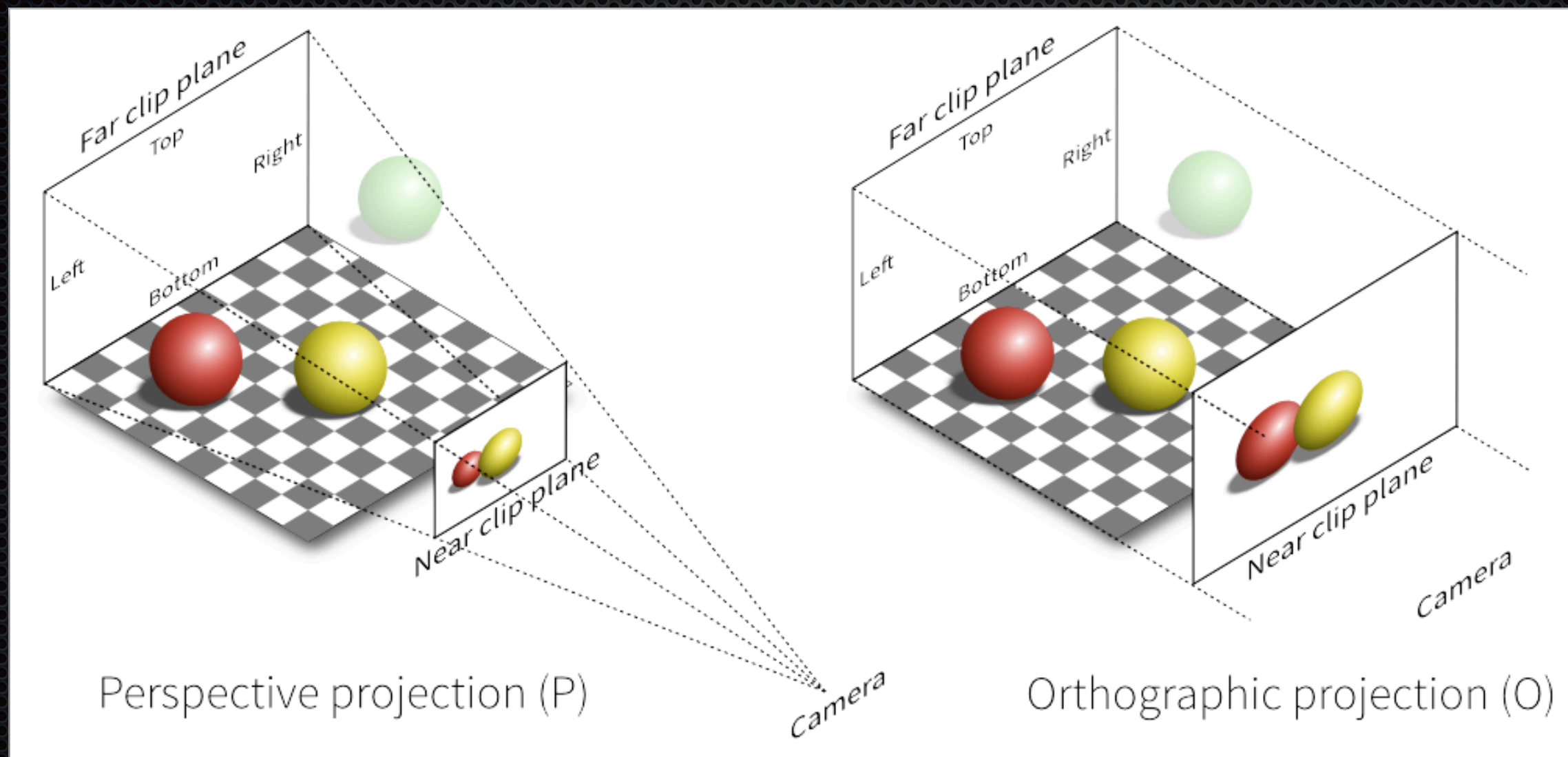


Orthographic Projection

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

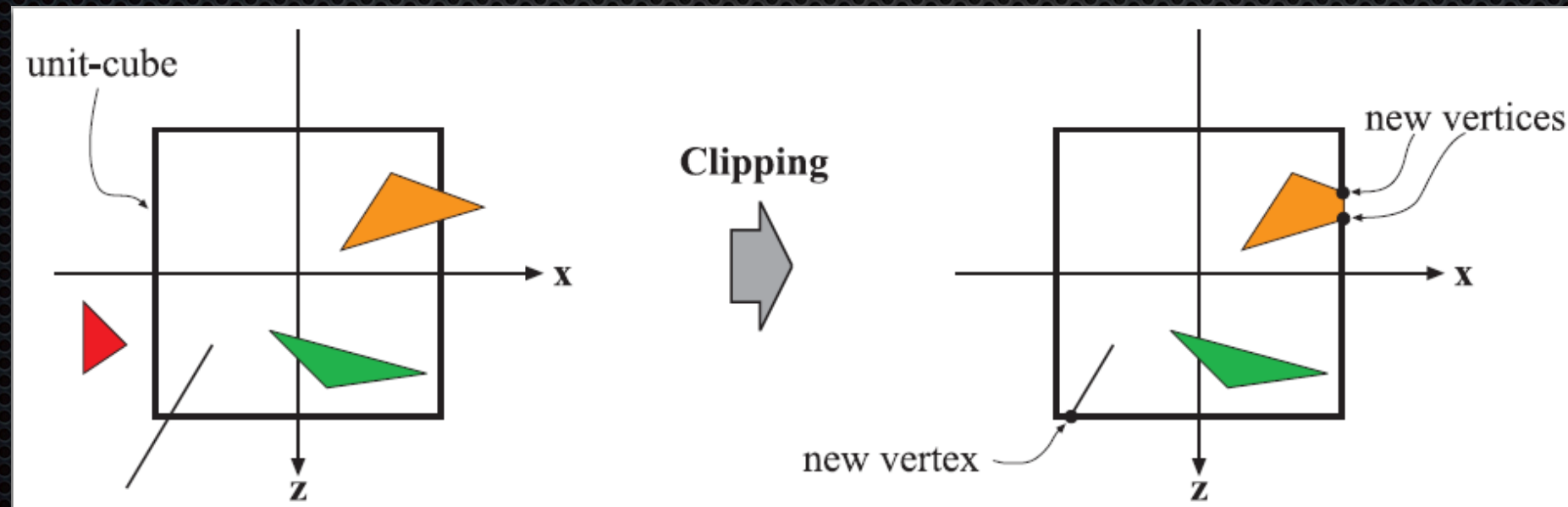


Perspective vs. Orthographic Projection



Clipping

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



Application



Geometry



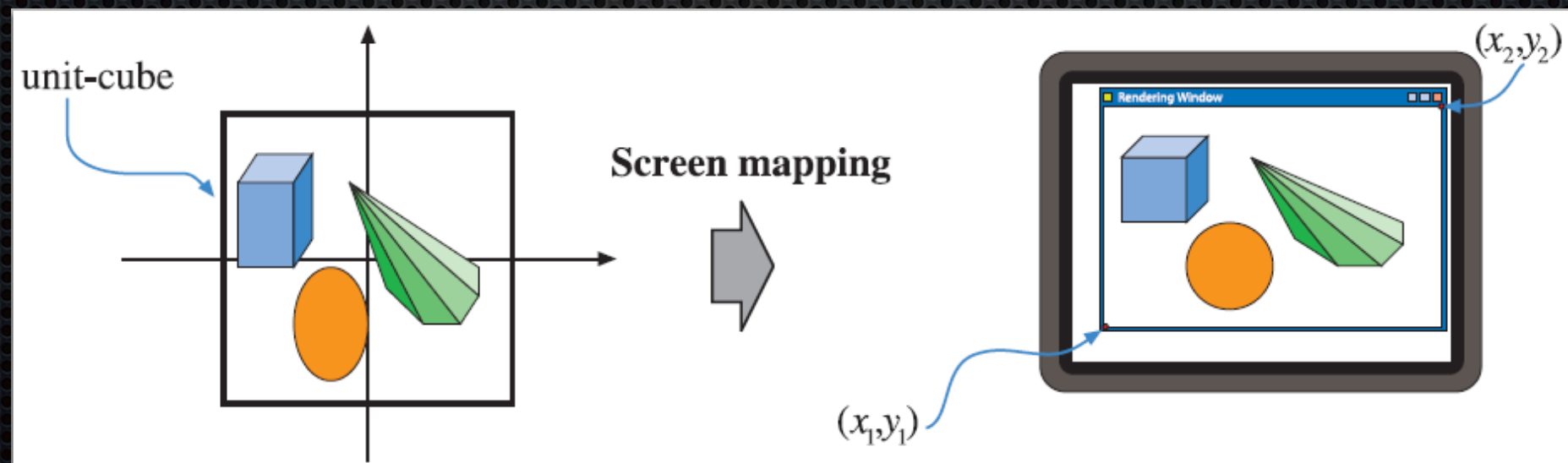
Rasterization



Screen

Screen Mapping

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



Application



Geometry



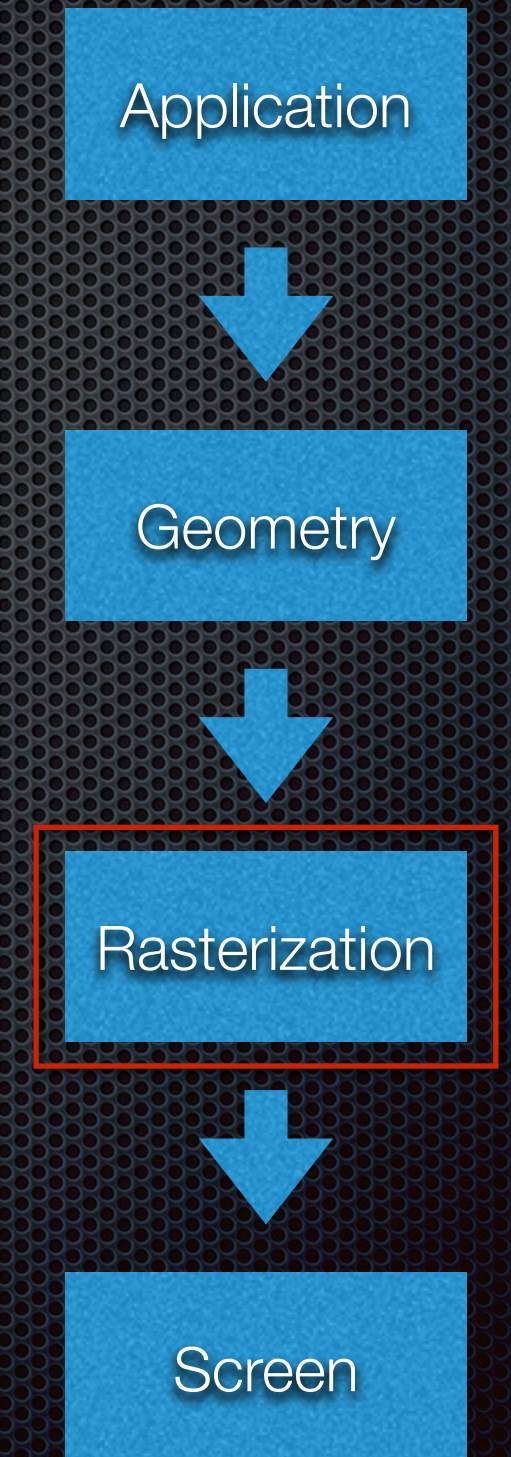
Rasterization



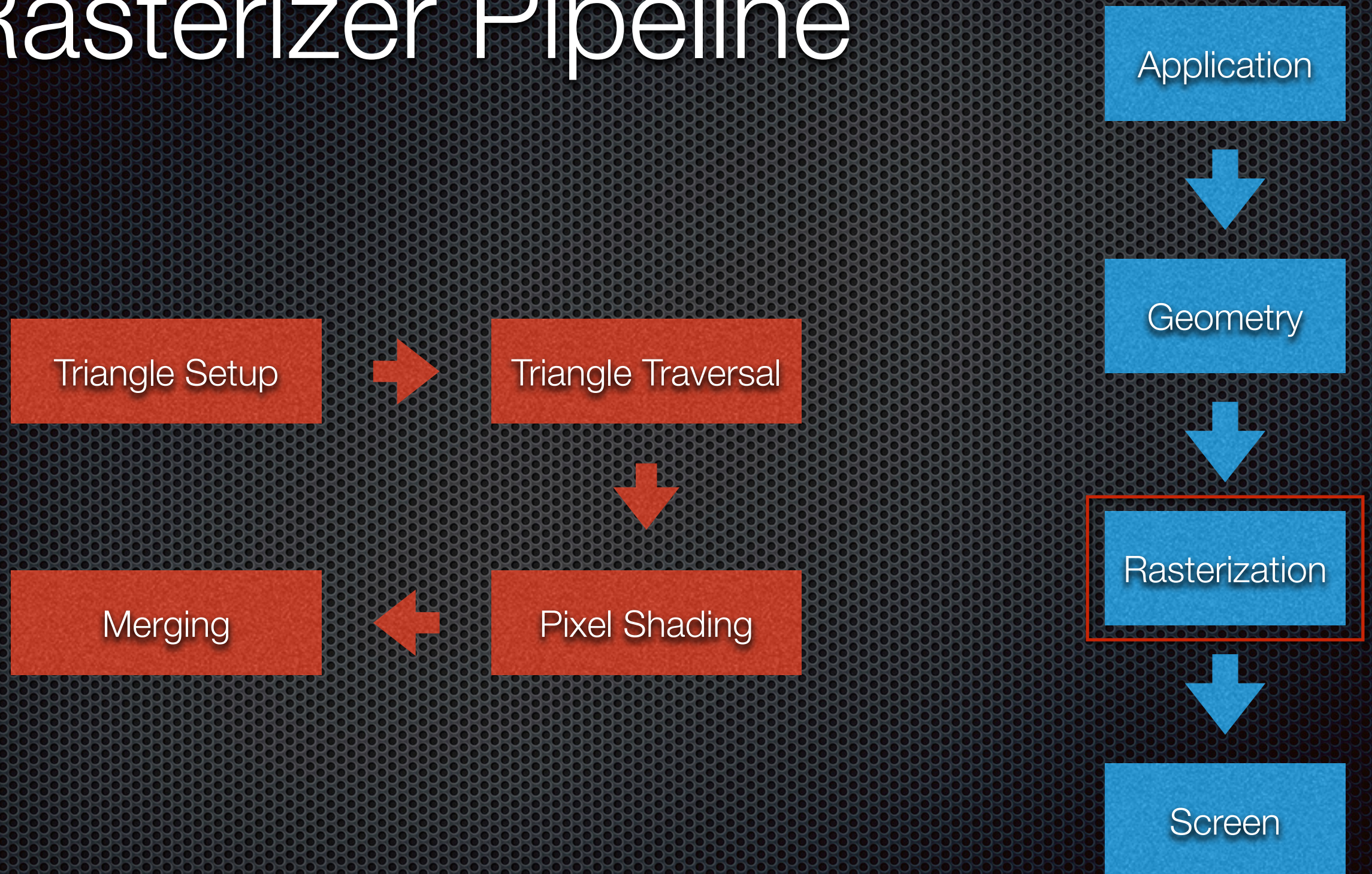
Screen

Rasterization

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

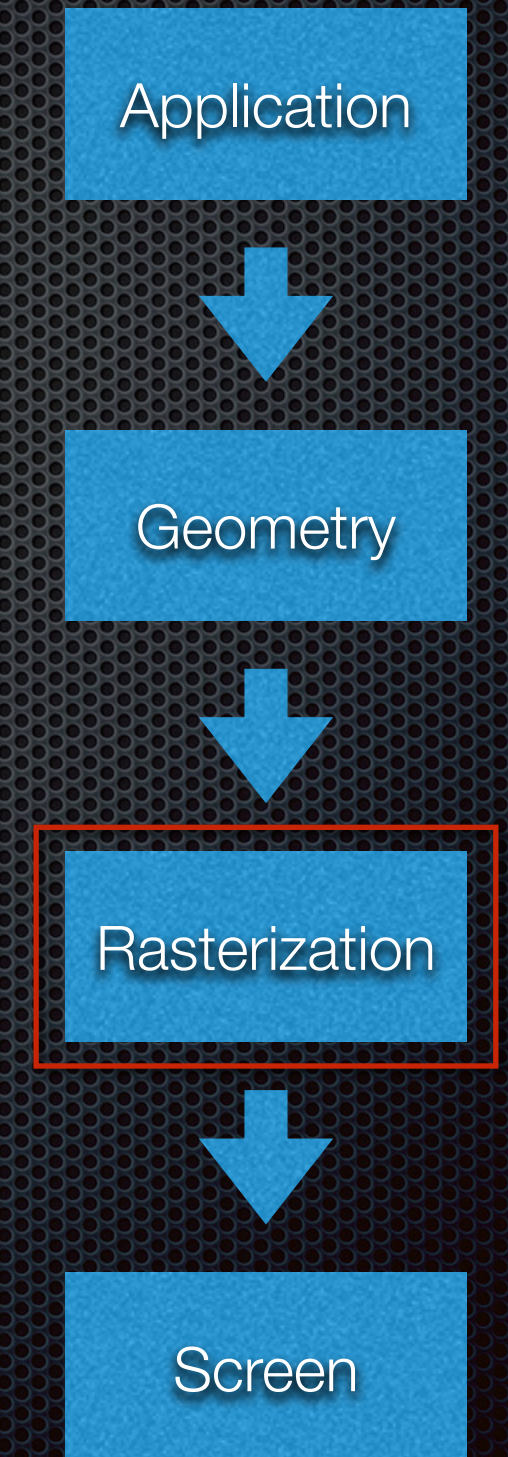


Rasterizer Pipeline

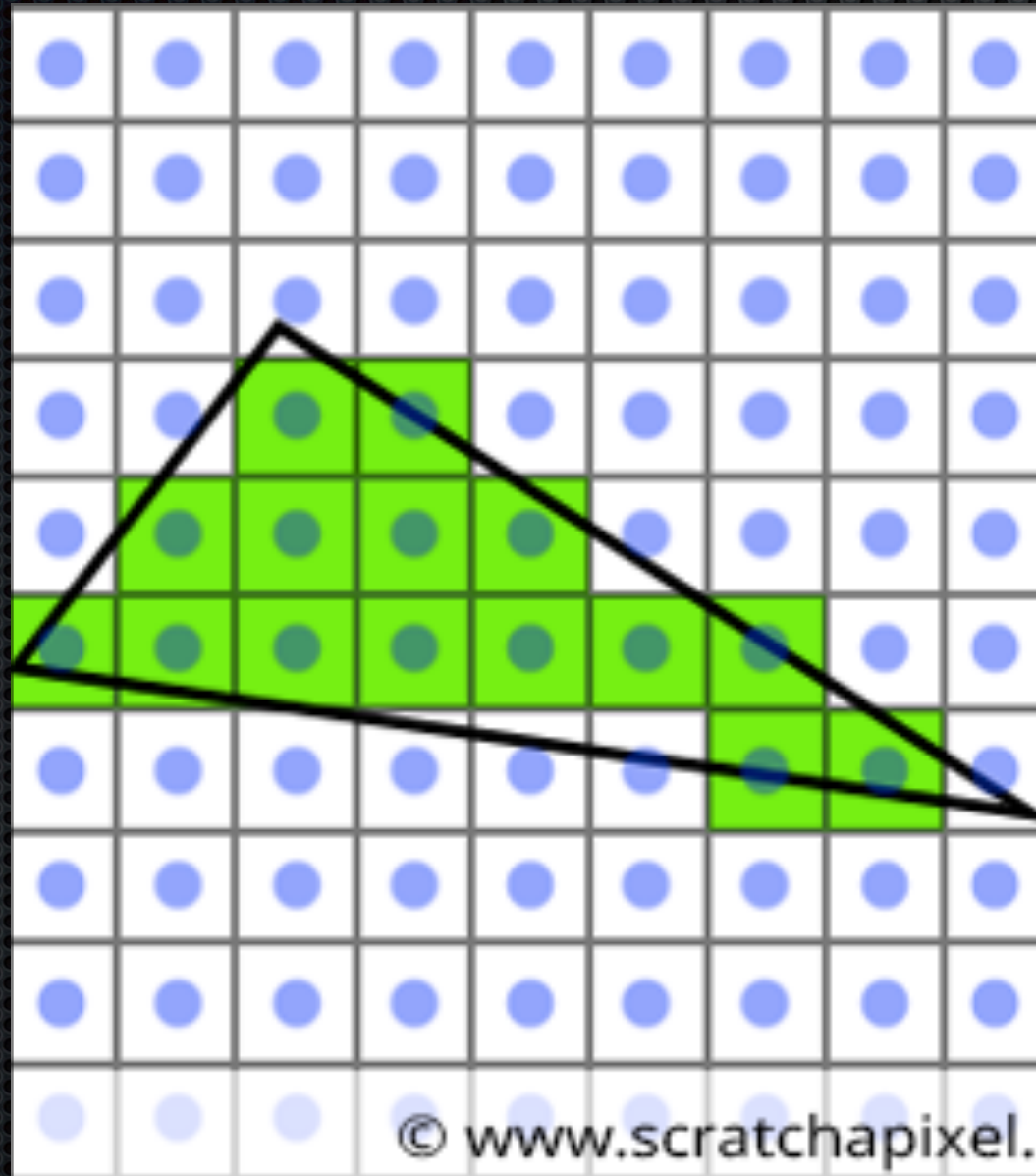


Triangle Setup

- ✦ primitives assembly
- ✦ calculation on the primitives
- ✦ face culling



Triangle Traversal



Application



Geometry



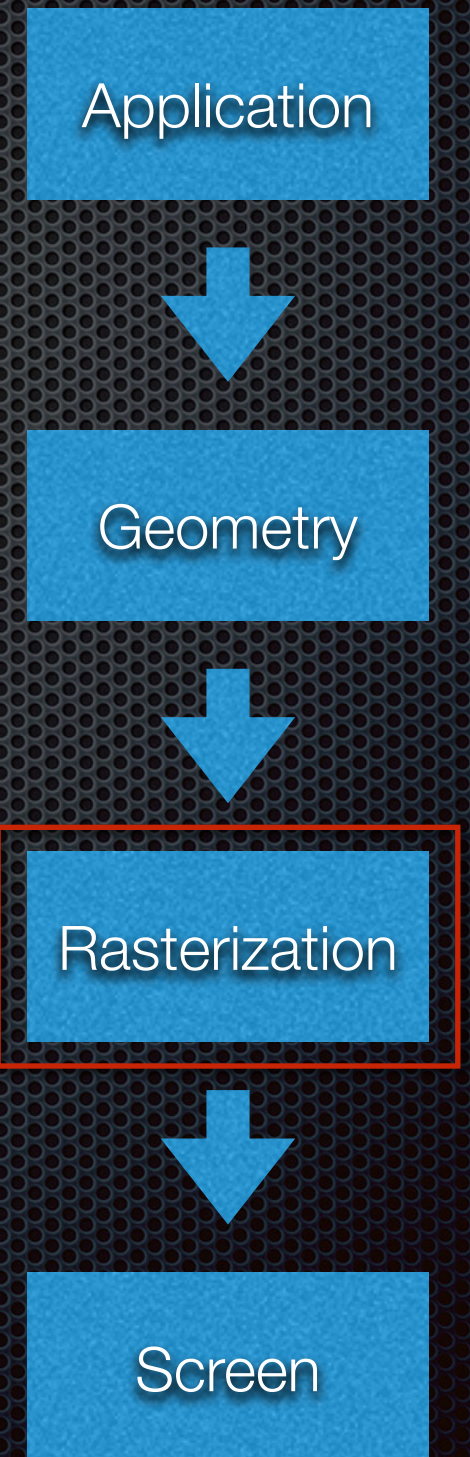
Rasterization



Screen

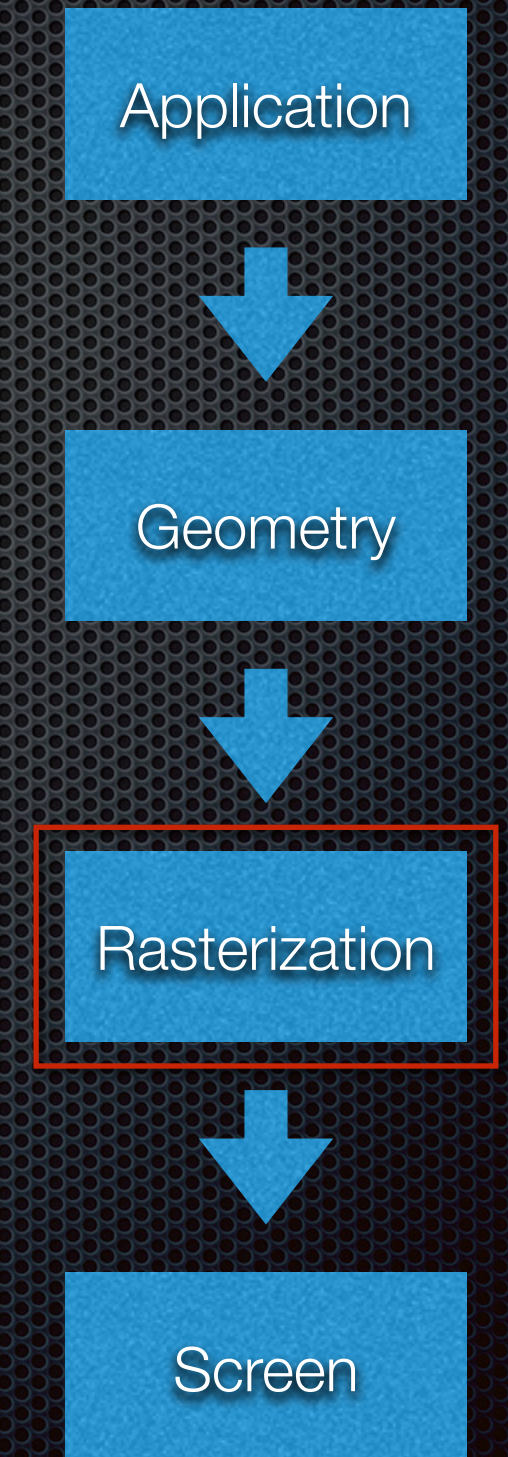
Pixel Shading

- ✦ final color
- ✦ texturing
- ✦ pixel lighting
- ✦ target is back buffer



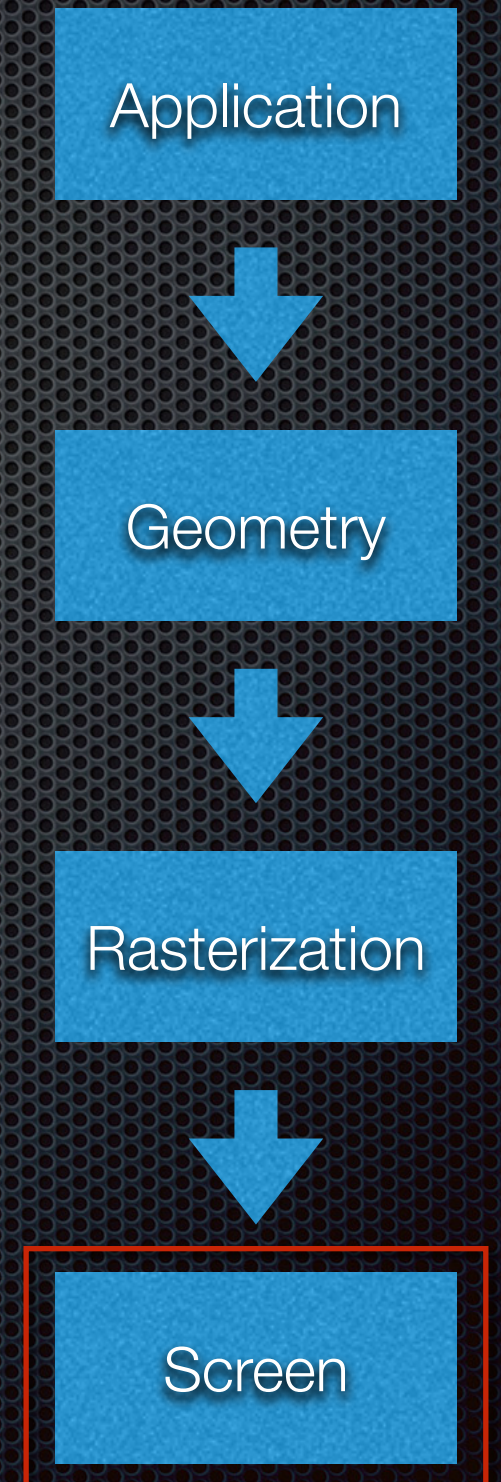
Merging

- ✦ Z-Test (Depth Buffer)
- ✦ alpha testing
- ✦ alpha blending
- ✦ stencil test (Stencil Buffer)



Screen

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



Front- and Backbuffer

- both are large memory dumps for saving pixel data
- size depends on resolution and pixel format
- frontbuffer
 - actual presented picture
- backbuffer
 - actual buffer graphic card is writing on
 - at least one

Vertical Synchronisation

- ✦ short VSync
- ✦ synchronize FPS of Application with FPS of output device
- ✦ preventing tearing
- ✦ save power



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

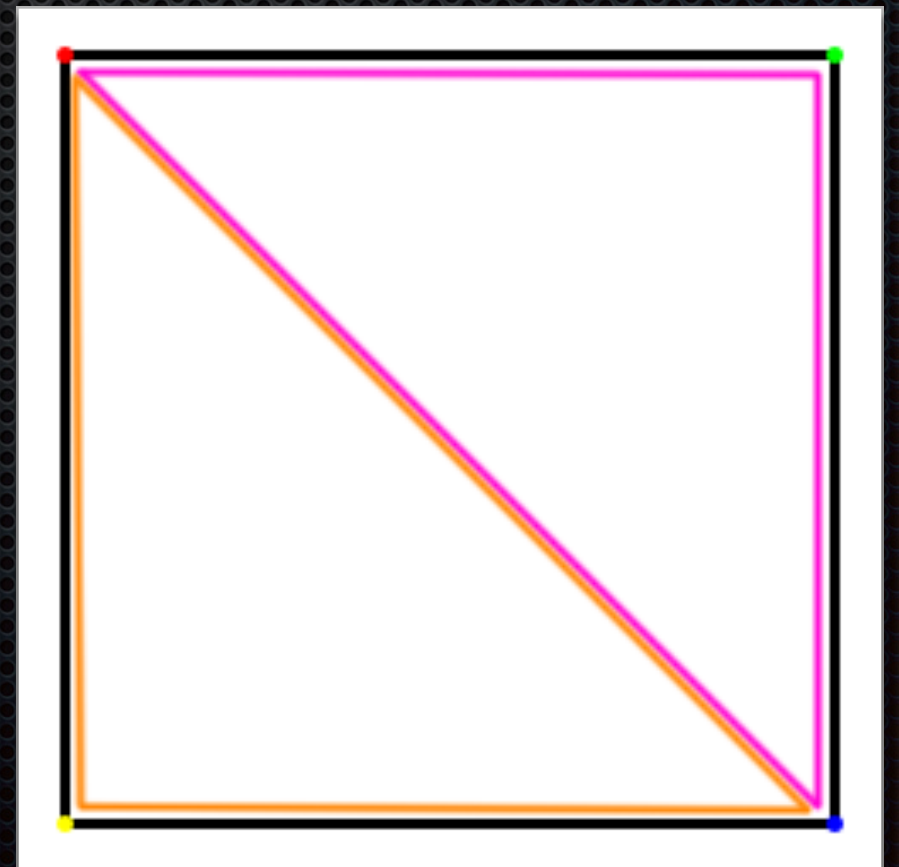
- indices refer to one vertex in vertexbuffer
- indices can refer to a vertex more than once, but
- only once for one primitive
- indices are saved sequentially
- primitives are
 - triangle
 - line
 - point

Create a Mesh

VertexBuffer



IndexBuffer



Coding Time

There are two types of people.

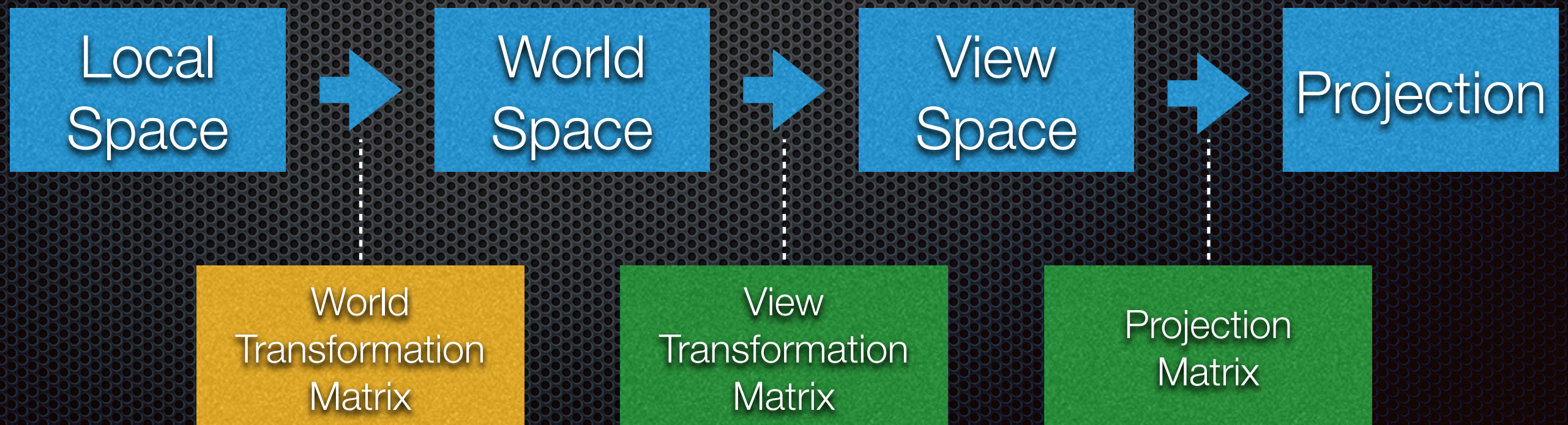
```
if (Condition)
{
    Statements
    /*
     *
     */
}
```

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

Programmers will know.

Transformation

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



DirectX & Mathematics

- standard data types
 - D3DVECTOR
 - D3DMATRIX
 - no helper methods
- D3DX...
 - is deprecated since Windows 8
 - included in DirectX SDK

DirectX & Mathematics

- DirectXMath
 - new library
 - SIMD friendly
 - difference between data storage and calculation
 - XMFLOAT... for storage
 - XMVECTOR / XMATRIX for calculation
 - DirectX Tool Kit (only 1.1) - 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

Coding Time

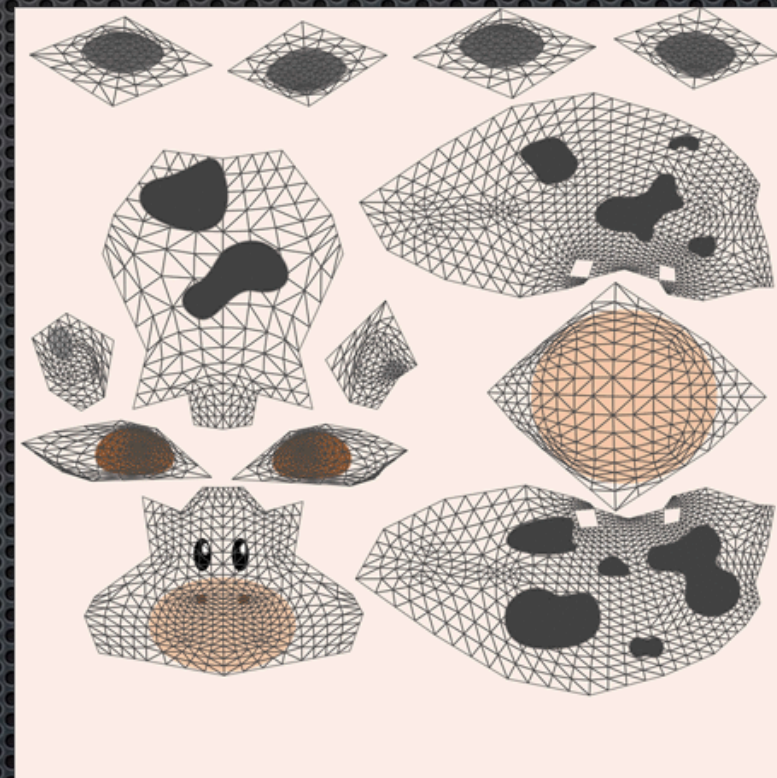
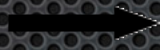
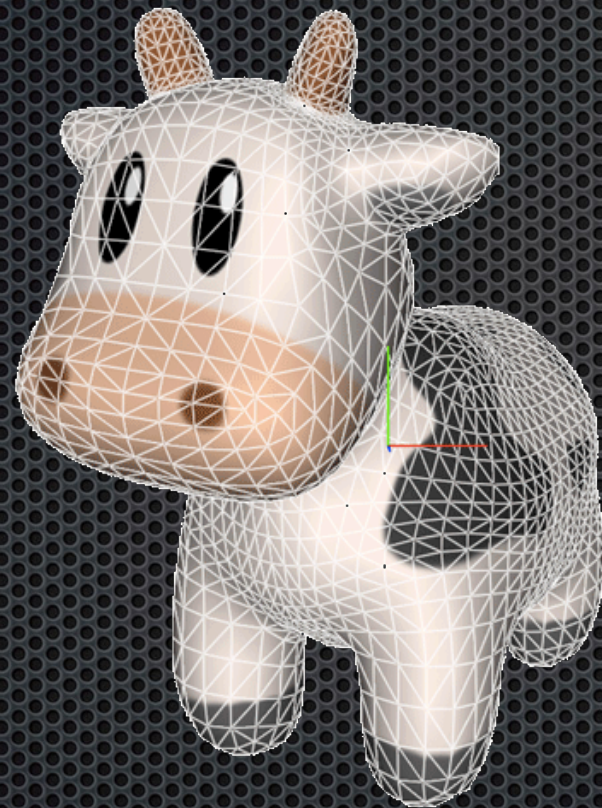
Hide and seek champion...

;

since 1958

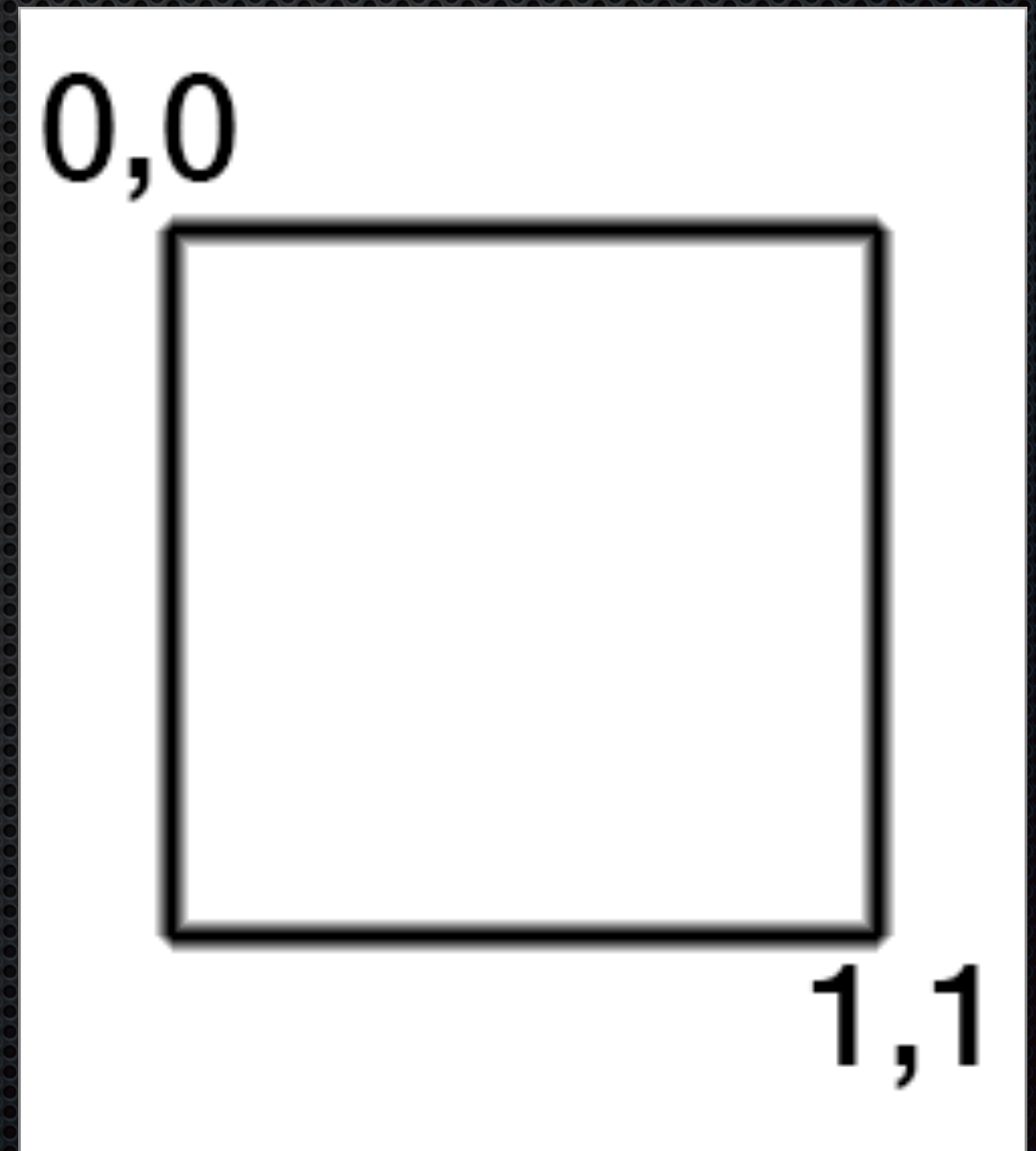
Texture Mapping

- needs for „skinning“ a mesh



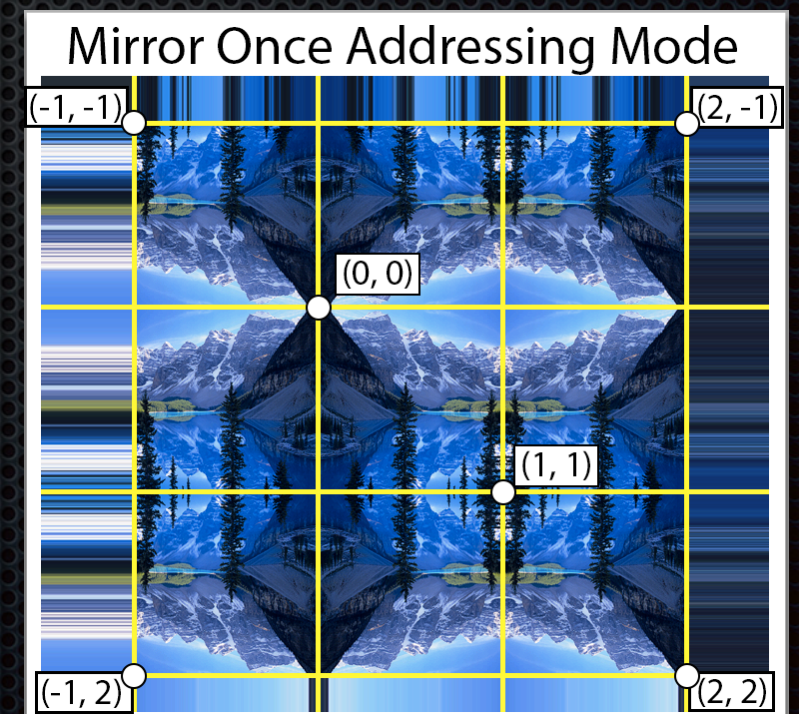
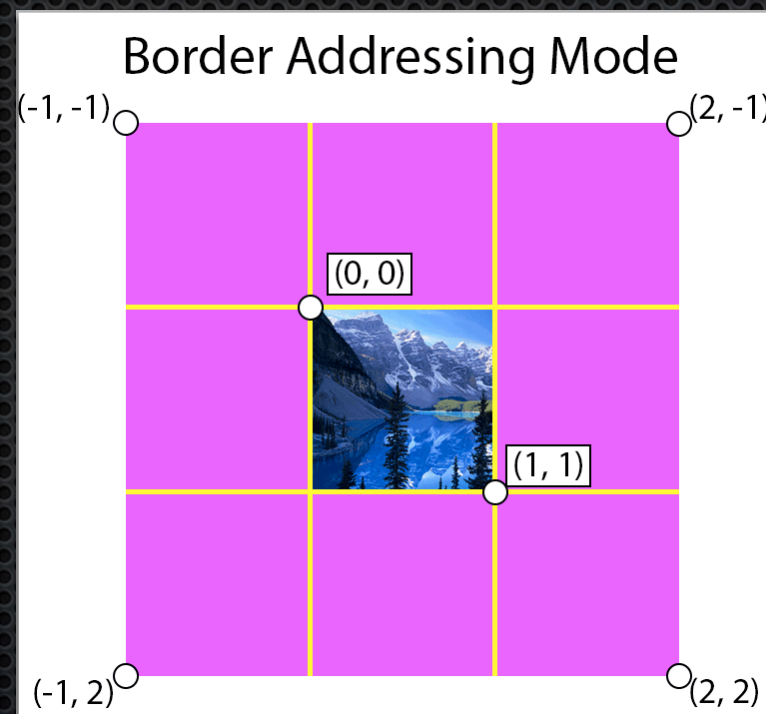
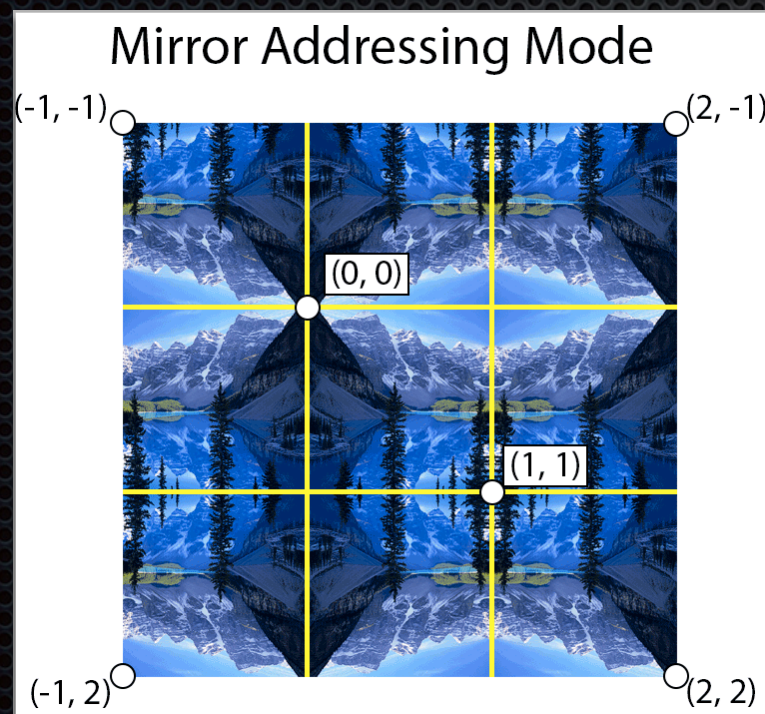
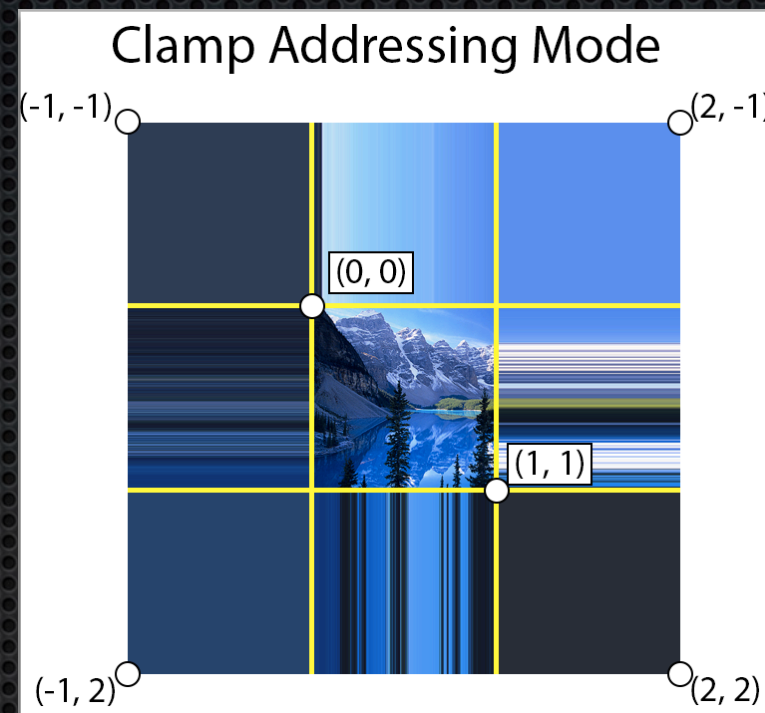
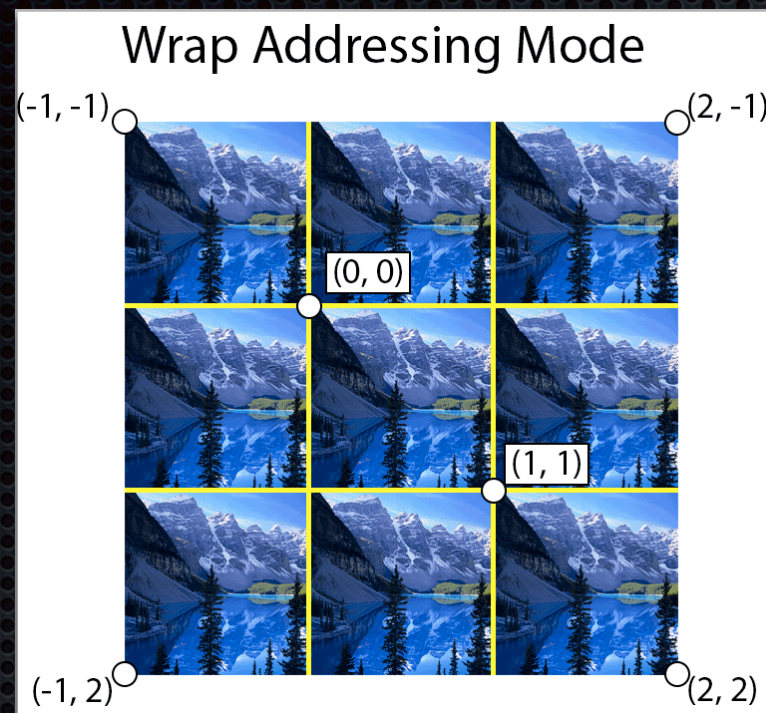
UV Mapping

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



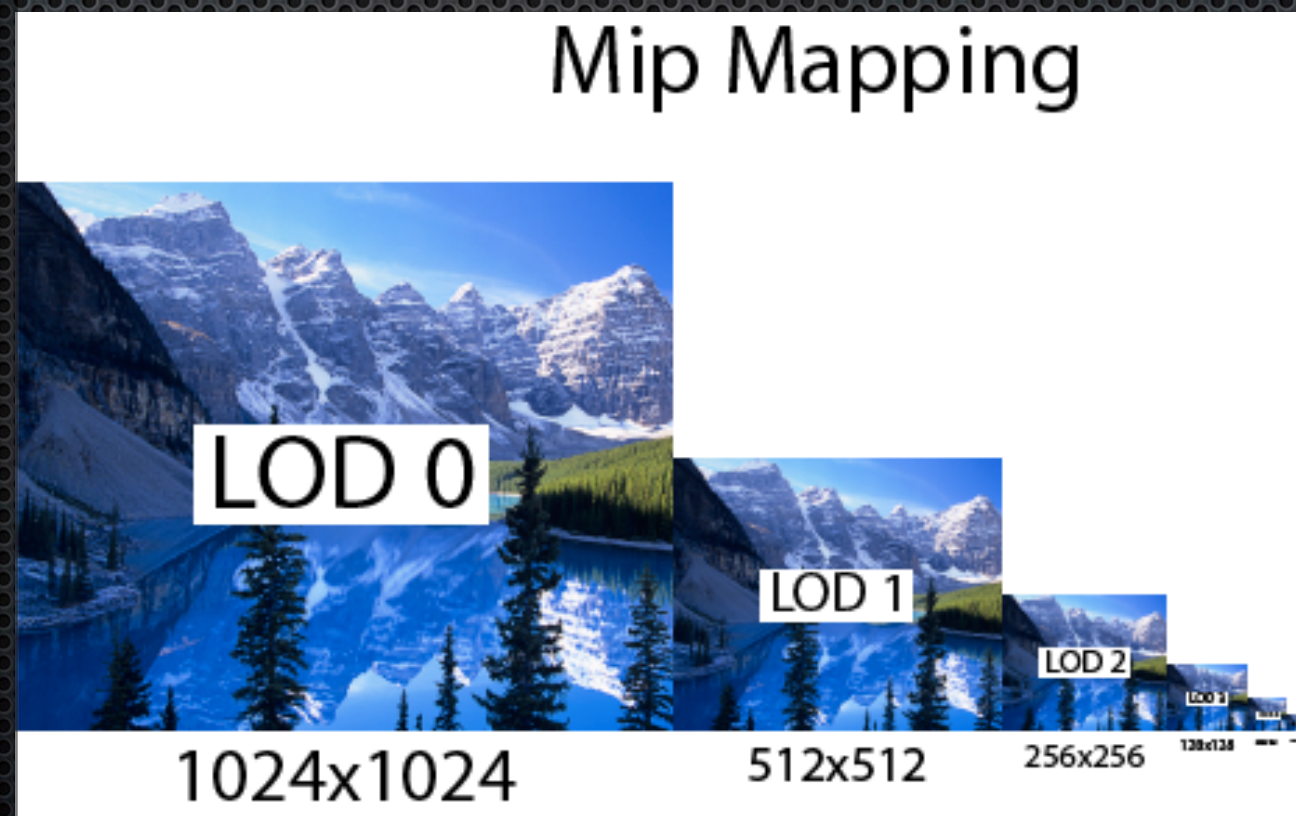
Sampler State I

- 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



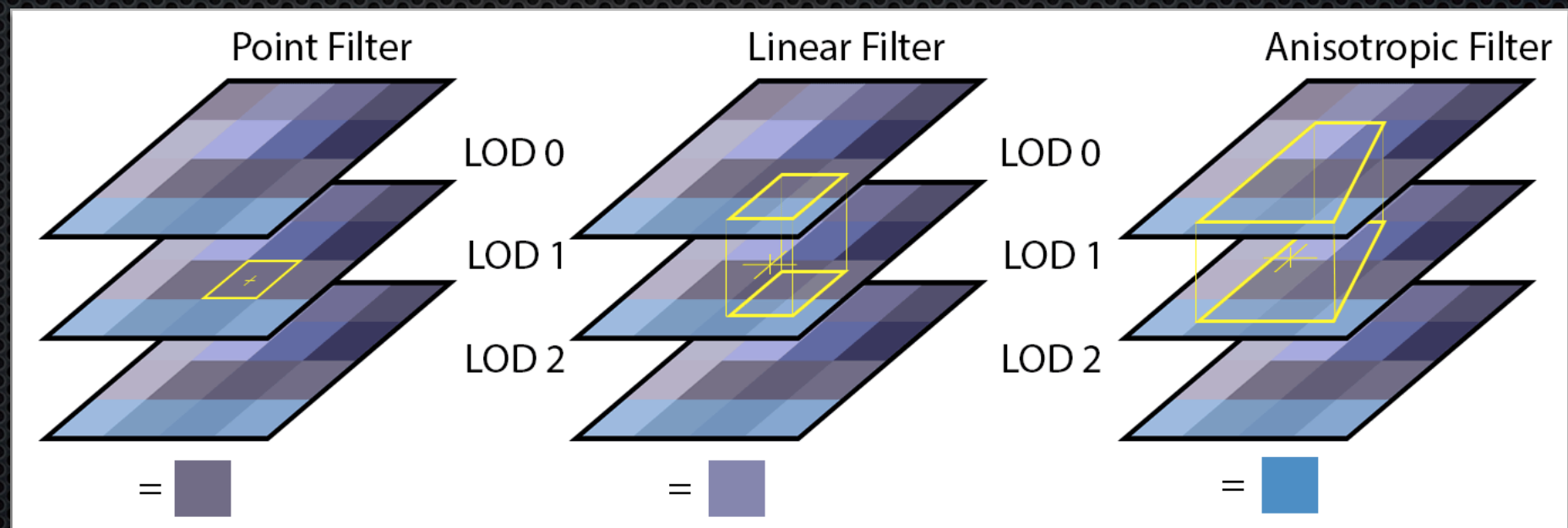
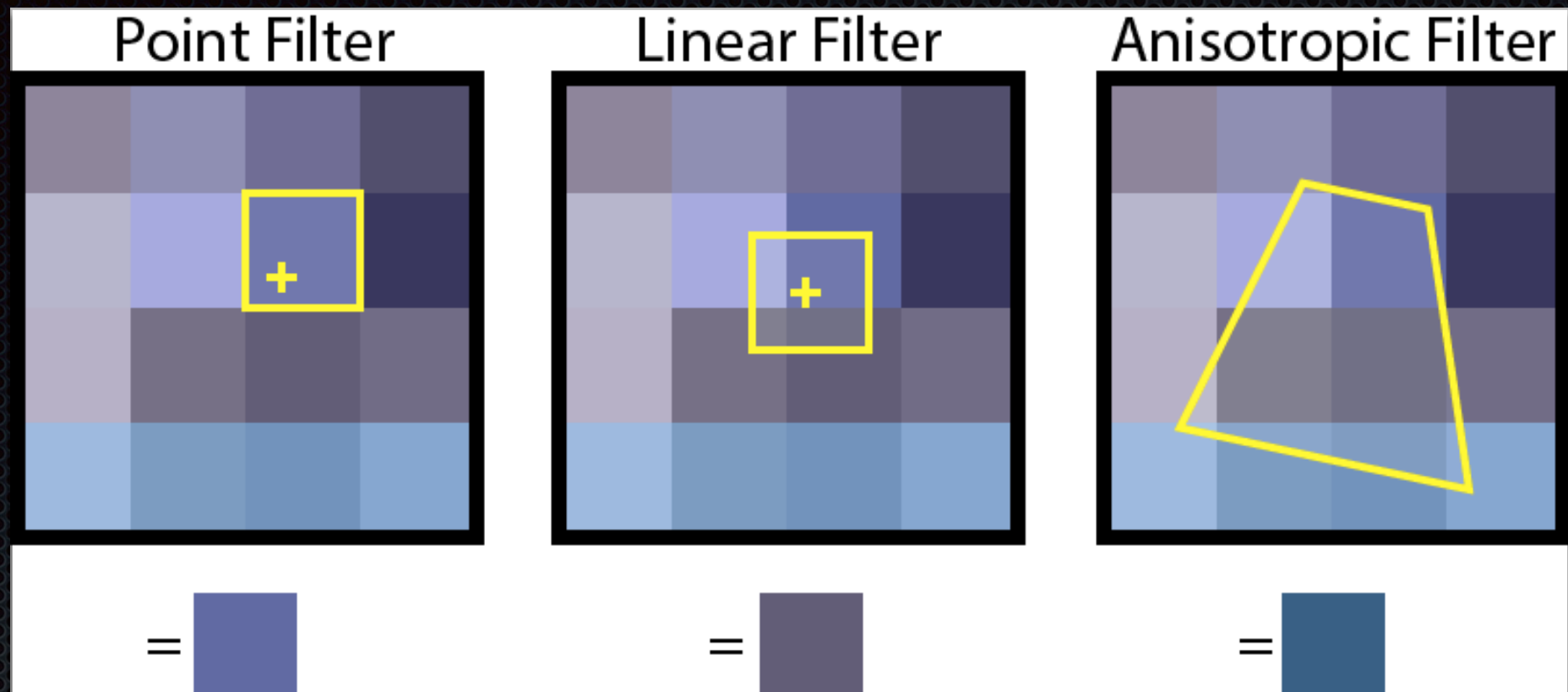
Sampler State II

- ✦ 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

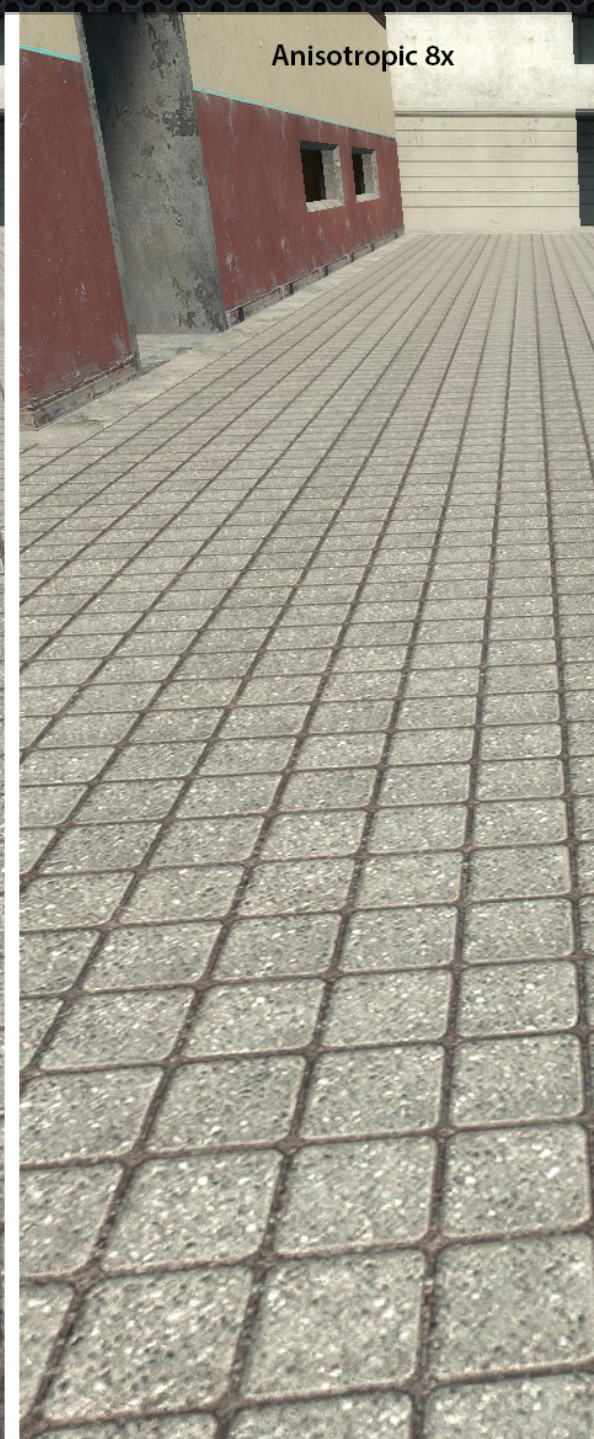
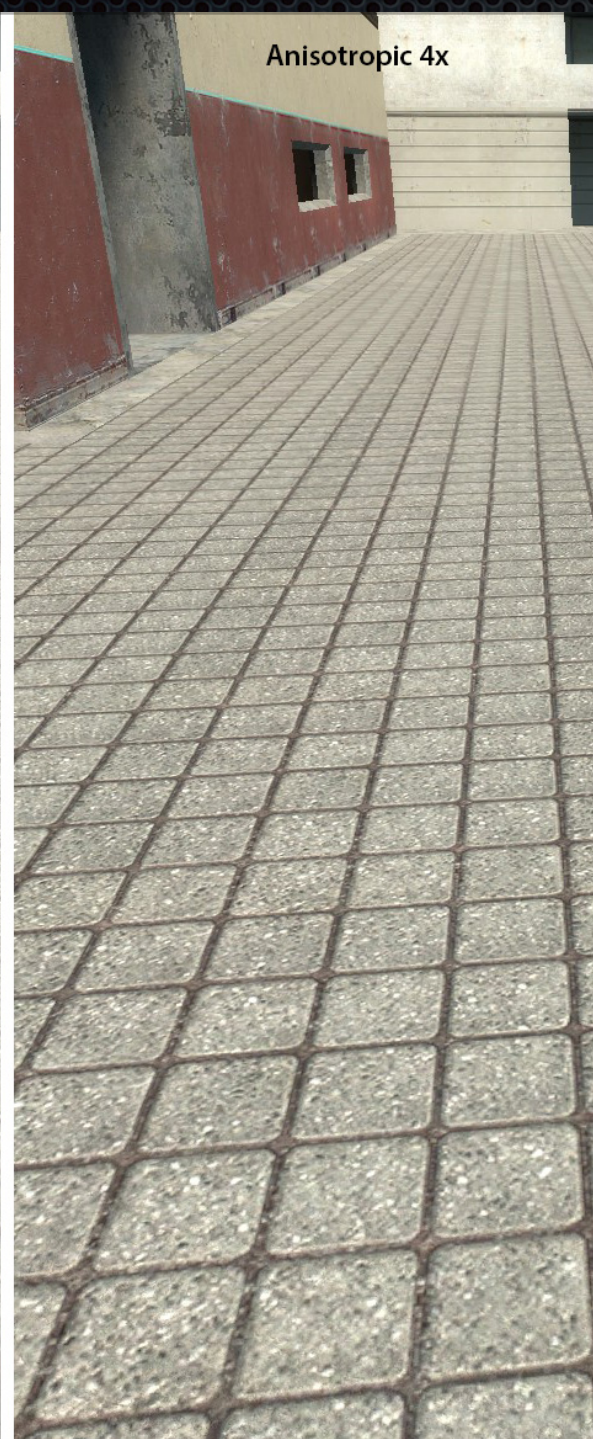


Linear



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.

Light

- ✦ creates more realism
- ✦ creates light and shadow
- ✦ can create more detail with normal mapping

Sources

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

Types

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

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

Light Source



L



N



R



V



Viewer



Surface



Coding Time

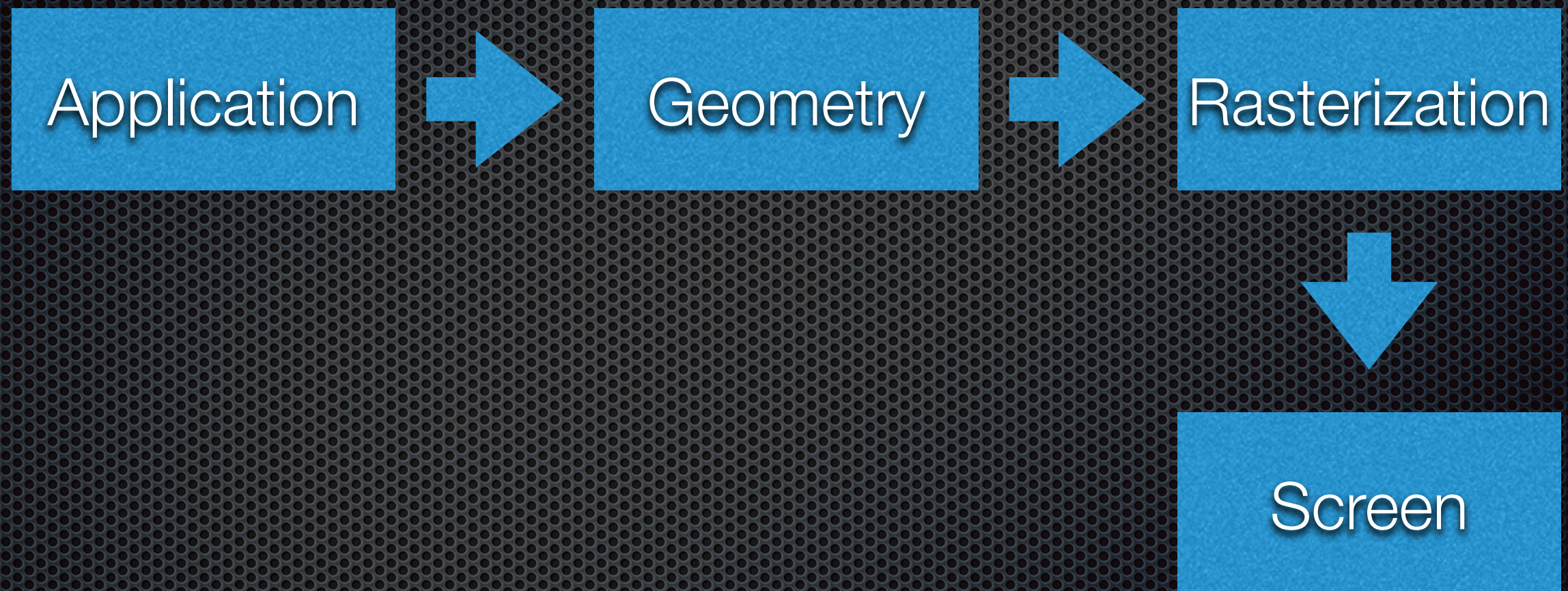
Have a great weekend
I hope your code behaves on Monday
the same way it did on Friday



STARECAT.COM

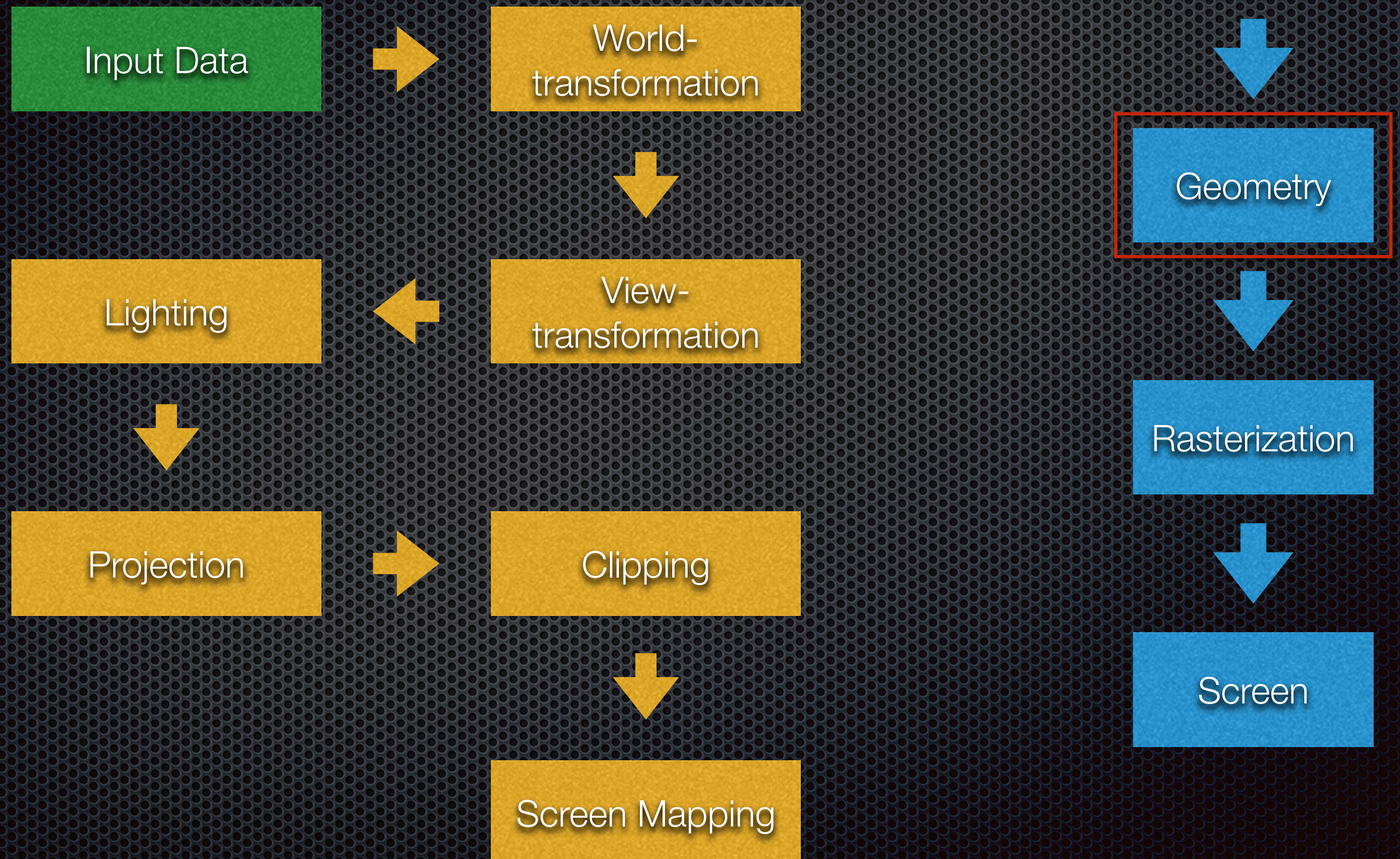
Programmable Pipeline

Structure



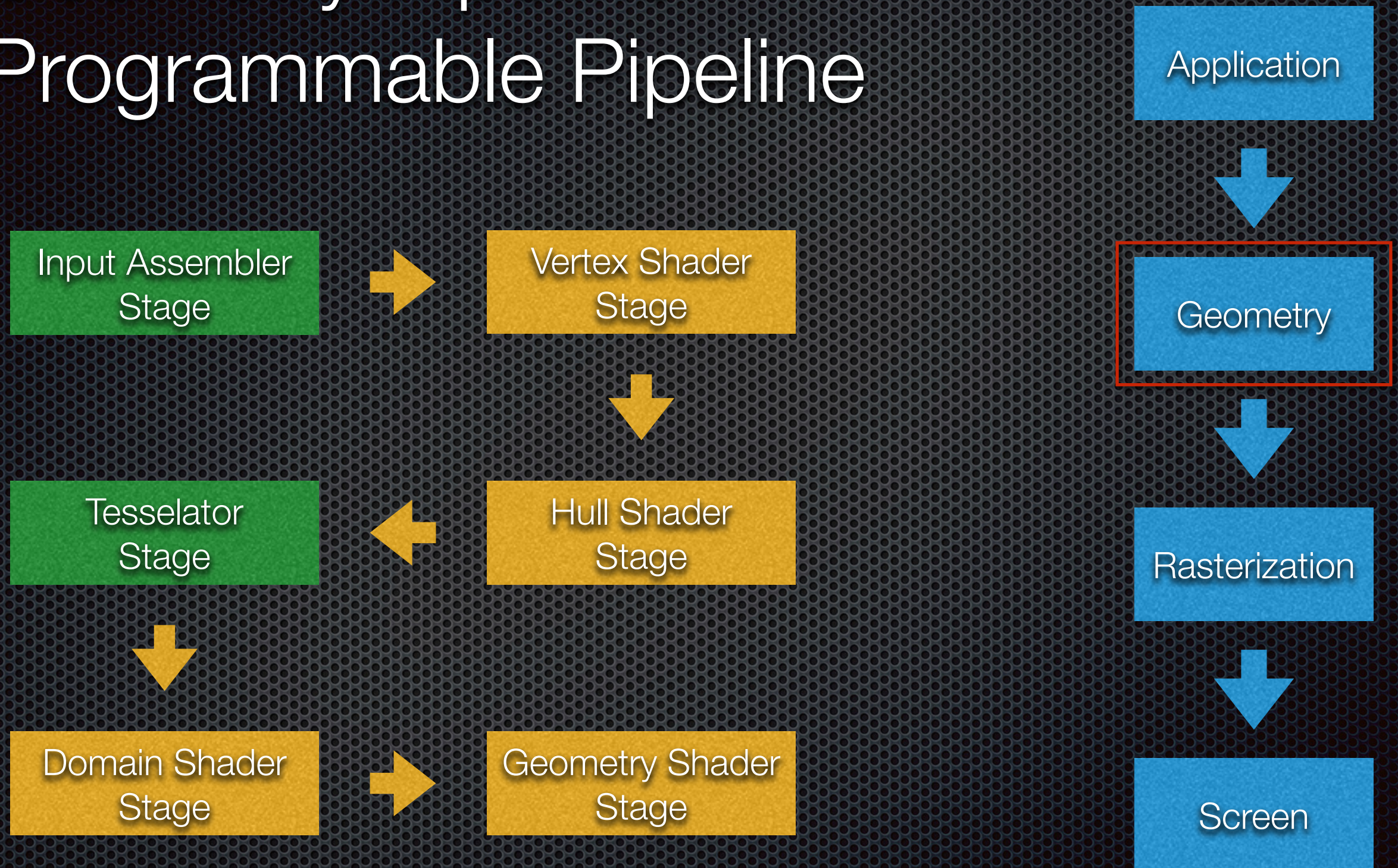
Geometry Pipeline

Fixed-Function Pipeline



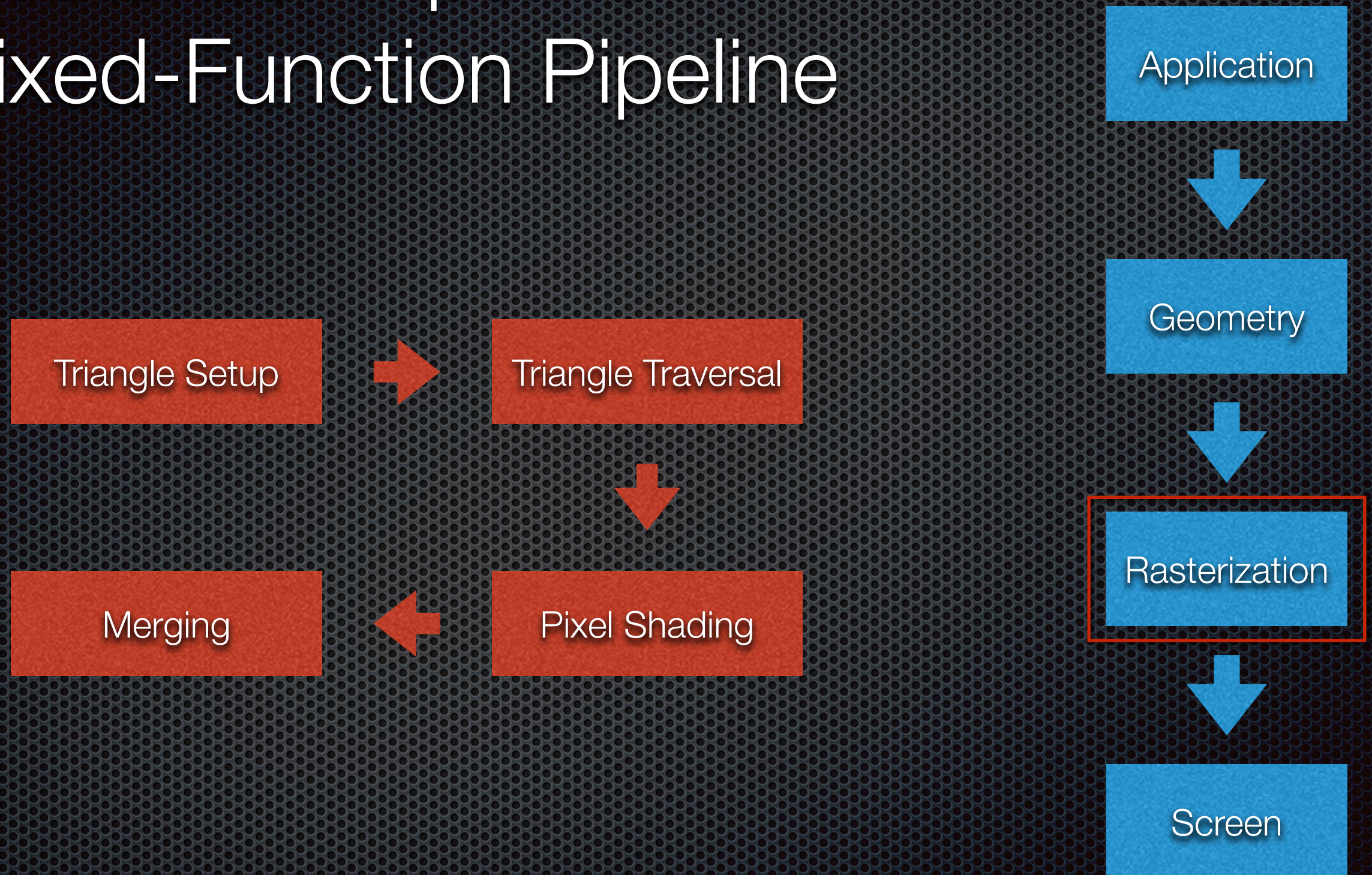
Geometry Pipeline

Programmable Pipeline



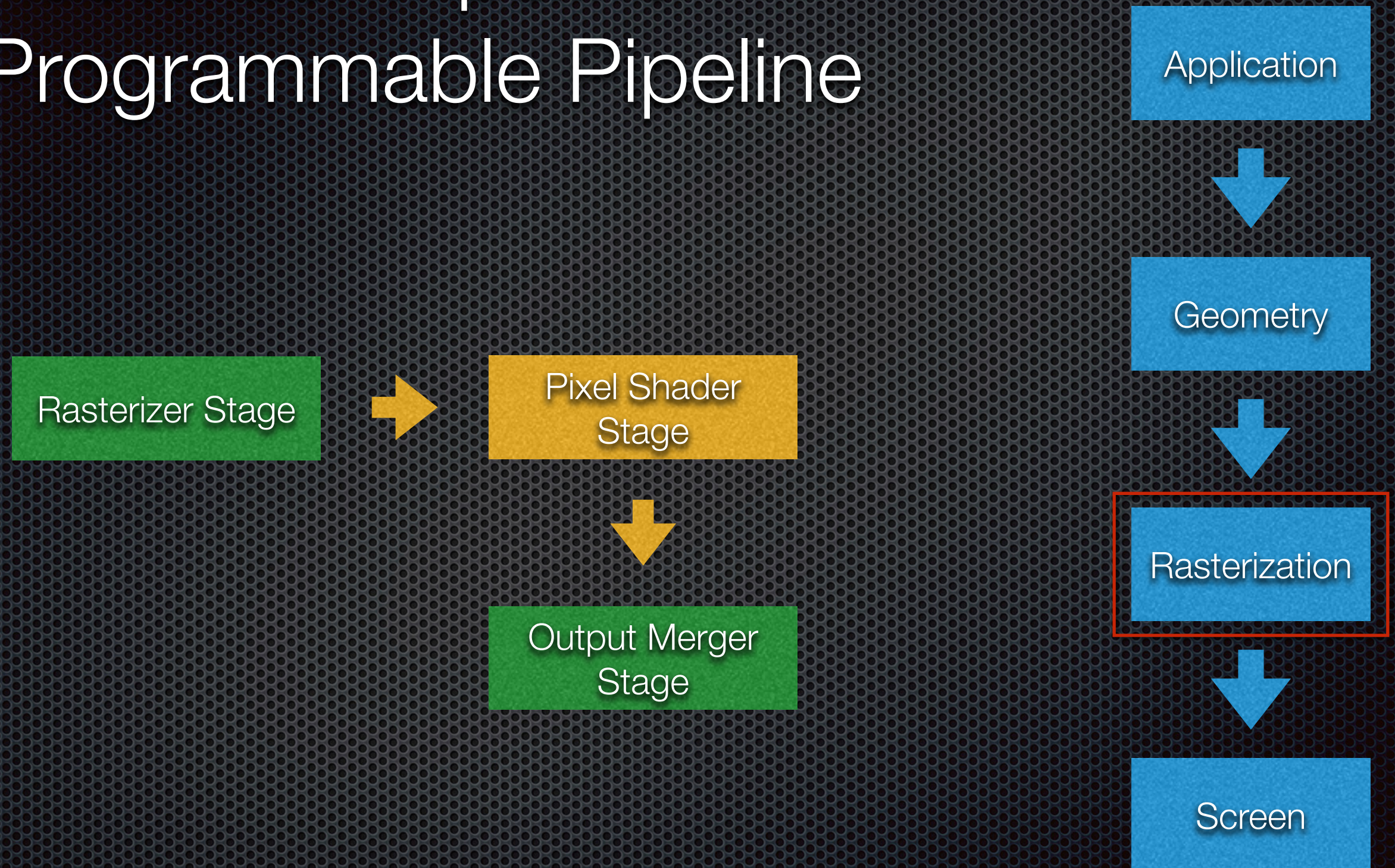
Rasterizer Pipeline

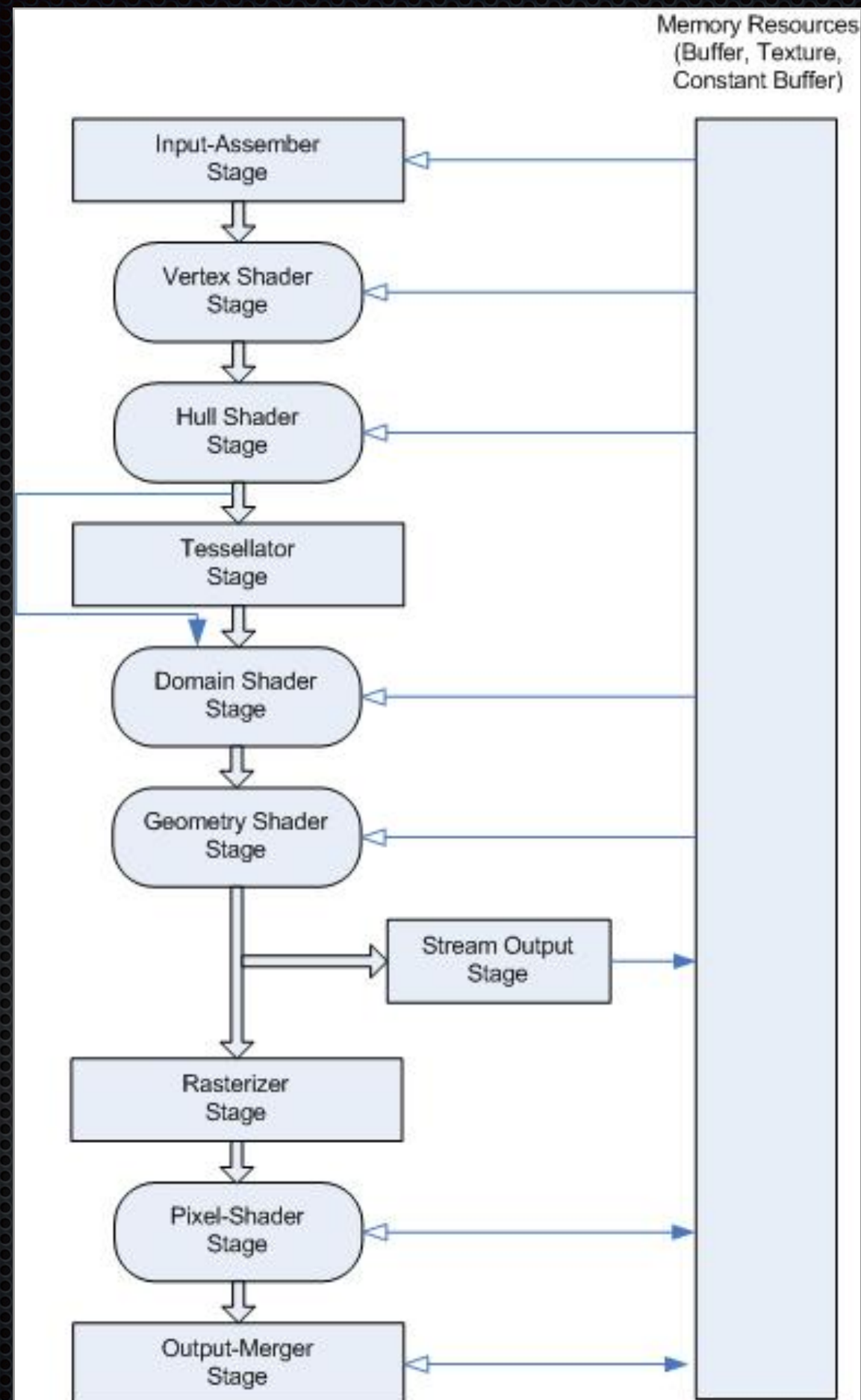
Fixed-Function Pipeline



Rasterizer Pipeline

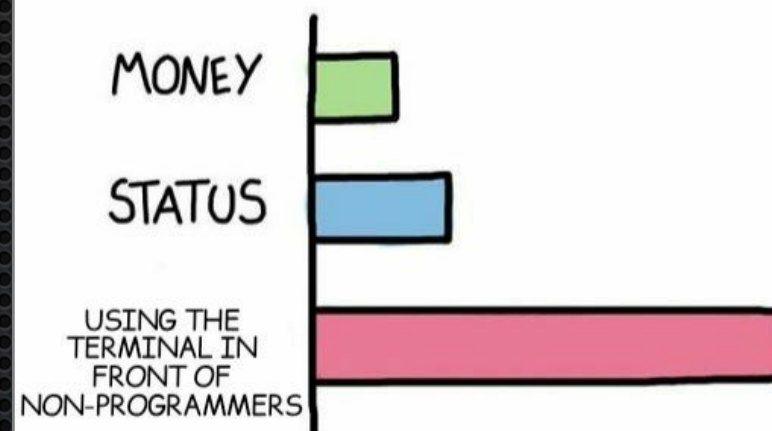
Programmable Pipeline





Coding Time

WHAT GIVES PEOPLE FEELINGS OF POWER



@iamnotanartist

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

Vertex Shader

- ✦ first shader in pipeline
- ✦ directly after input assembler
- ✦ performs operations on every single vertex

Pixel Shader

- ✦ last shader in pipeline
- ✦ calculate final pixel color and write it to the back buffer
- ✦ can calculate a depth value and write it to the depth buffer

Geometry Shader

- ✦ optional shader
- ✦ deals with whole primitives like triangles, lines or points
- ✦ can discard or create one or more primitives
- ✦ e.g. for creating point sprites or volumes

Hull/Domain Shader

- ✦ optional
- ✦ subdivides meshes into smaller primitives (Tessellation)
- ✦ normally based on mathematical functions, e.g. camera distance or edge length

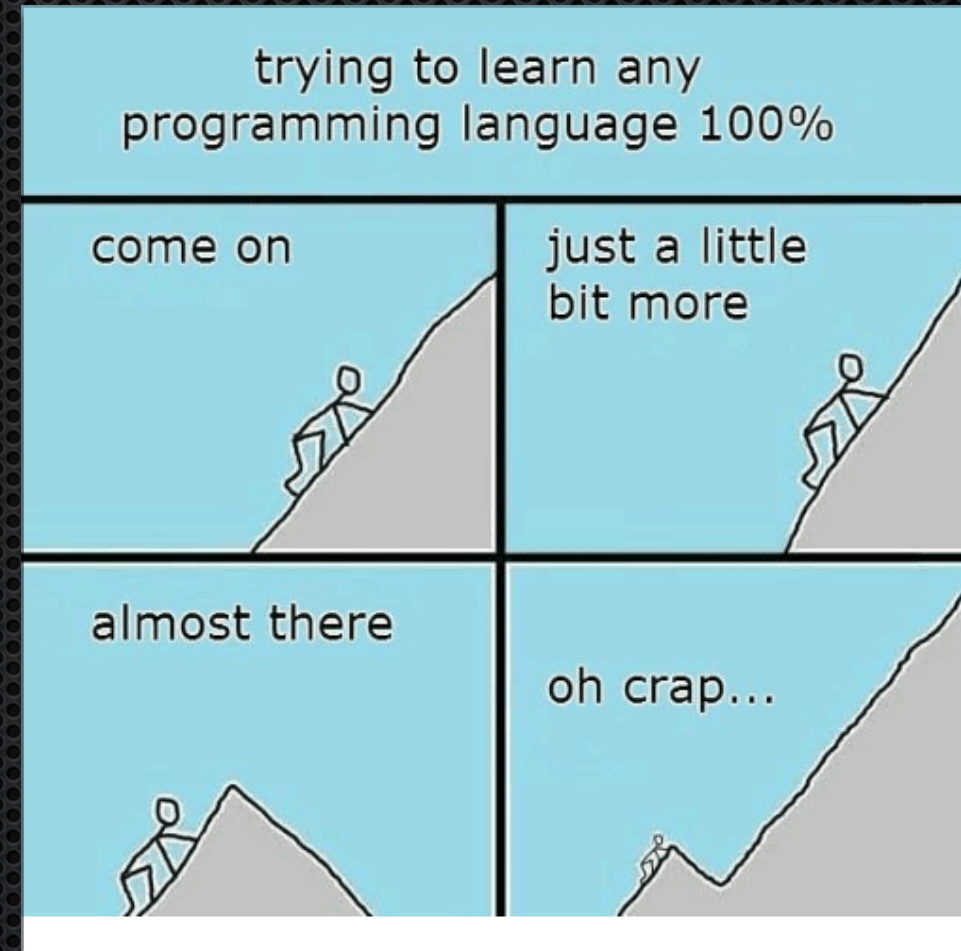
Compute Shader

- ✦ optional
- ✦ used for additional calculation
- ✦ not limited to graphics pipeline
- ✦ used for heavy float calculation

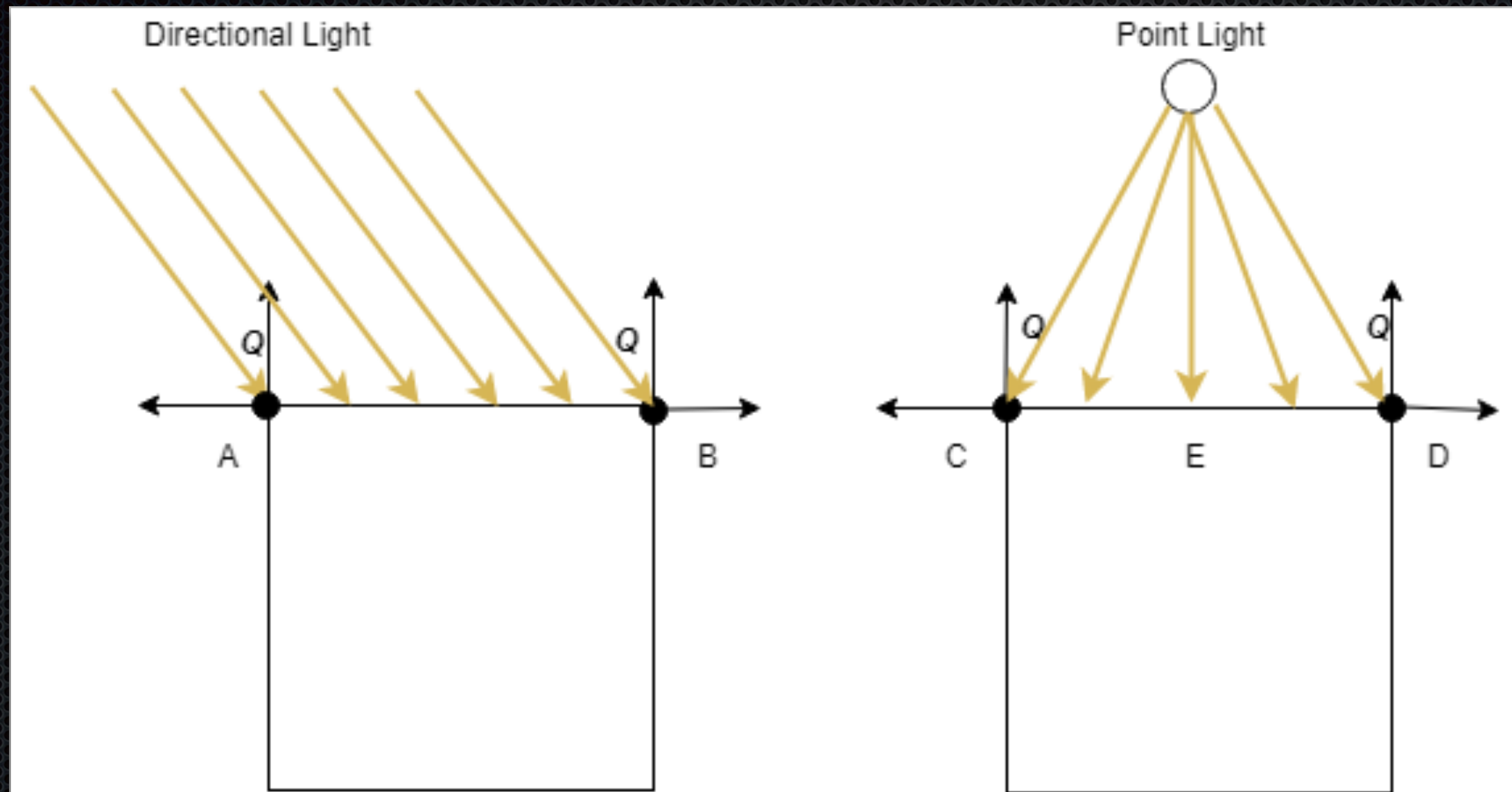
Shader Resources

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

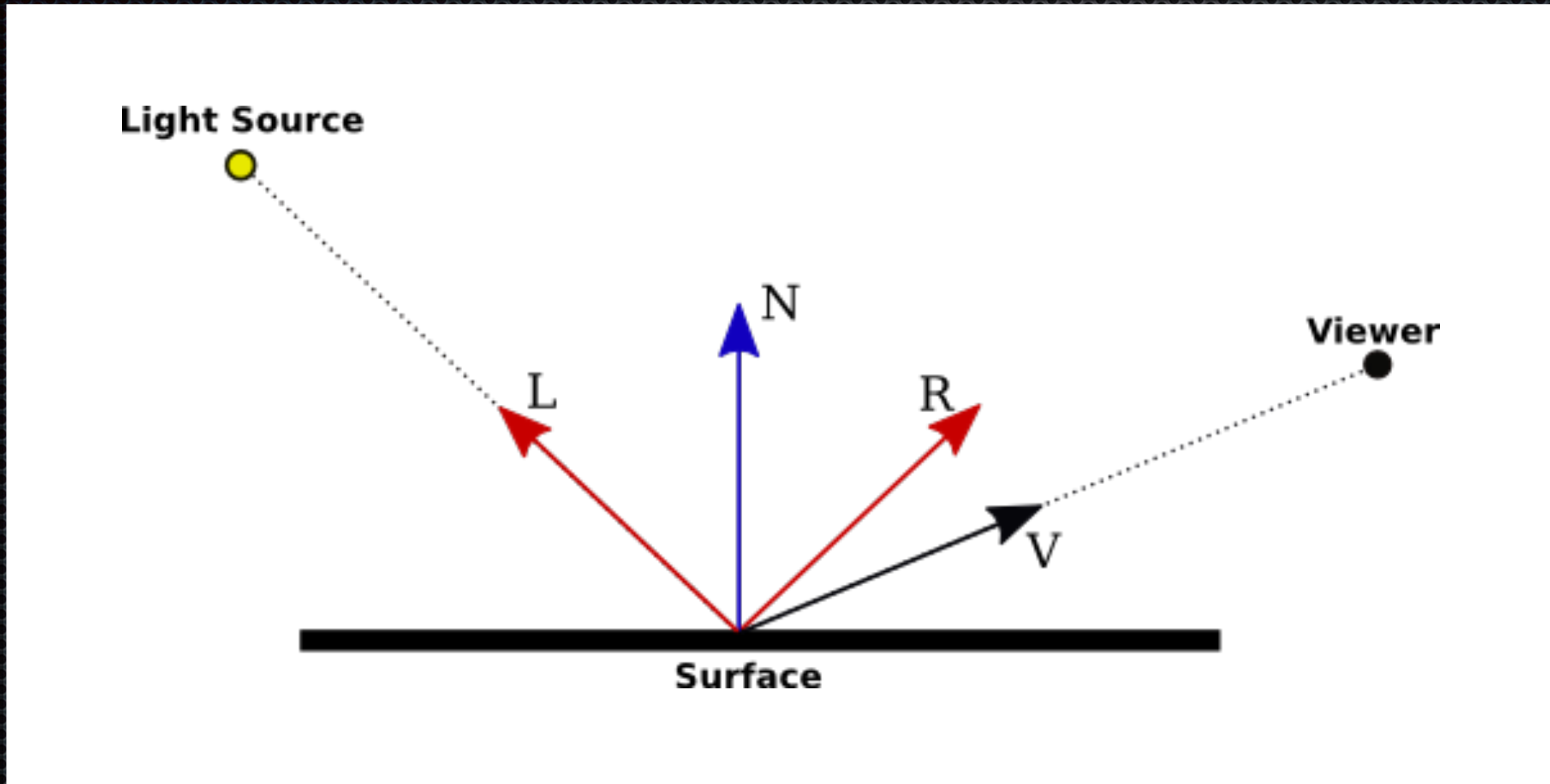
Coding Time



Light Source



Light Calculation



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

Coding Time



Comments to
describe the
program



Comments
to
temporarily
remove
part of
code

Unity Render Pipeline

Available Pipelines

- ✦ Built-In Render Pipeline
- ✦ Scriptable Render Pipeline
 - ✦ Universal Render Pipeline (URP, old LRP)
 - ✦ High Definition Render Pipeline (HDRP)

Writing Shaders

- ✦ Language: ShaderLab & CgFx/HLSL
- ✦ Fixed Function Shaders
- ✦ Vertex & Pixel (Fragment) Shaders
- ✦ Surface Shaders

Rendering Paths

- ✦ Forward Rendering
- ✦ Deferred Shading
- ✦ Deferred Lighting (legacy)
- ✦ Vertex Lit (legacy)

Coding Time

There are two types of people

```
HW.cpp
1  #include <iostream>
2
3  int main() {
4      std::cout << "Hello World!";
5      return 0;
6  };
7
```

```
HW.cpp
1  #include <iostream>
2
3  int main() {
4      std::cout << "Hello World!";
5      return 0;
6  };
7
```

VIA 9GAG.COM