Introducing Graphics Architecture

Chakrit Watcharopas

Introducing Graphics Architecture

3D rendering

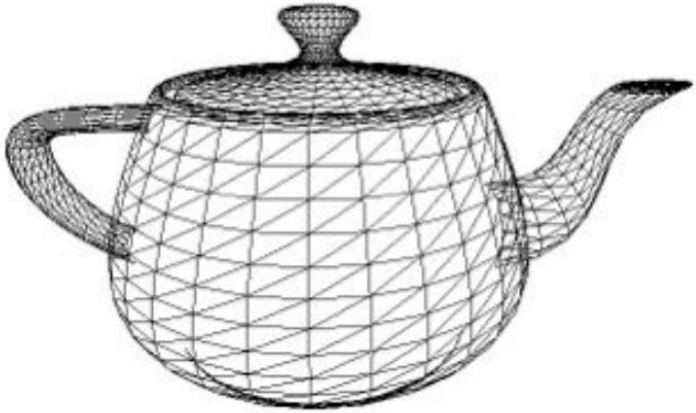


Image credit: Henrik Wann Jensen

Model of a scene:

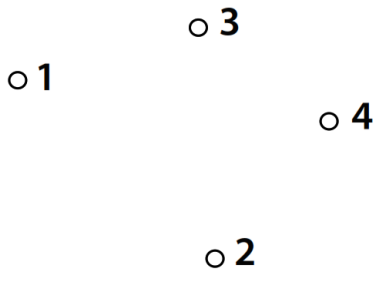
3D surface geometry (e.g., triangle mesh)
surface materials
lights
camera

Image

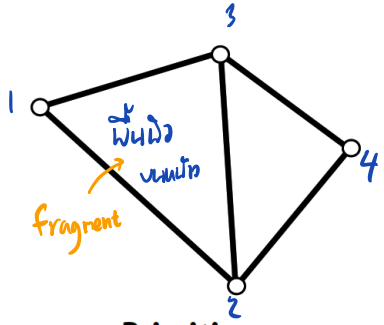
เราจะเขียนโปรแกรม
ให้มันหมุนมุมกล้อง

How does each triangle contribute to each pixel in the image?

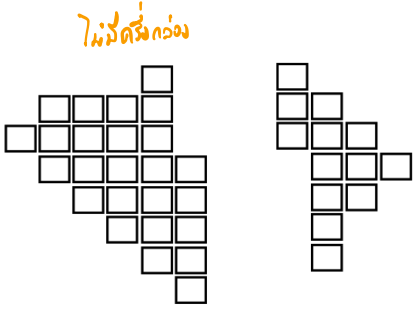
Real-time graphics pipeline (entities)



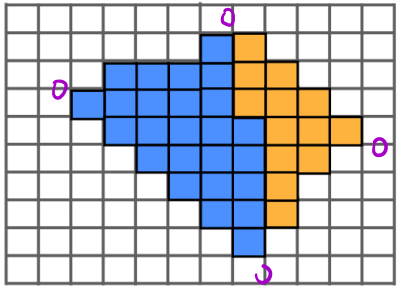
Vertices



Primitives
(triangles, points, lines)



Fragments



Pixels

Real-time graphics pipeline (operations)

ทางเข้าท่อ

Vertices

Vertex Generation

Vertex stream

Vertex Processing

Vertex stream

Primitives

Primitive Generation

Primitive stream

Primitive Processing

Primitive stream

Fragments

Fragment Generation (Rasterization)

Fragment stream

Fragment Processing

Fragment stream

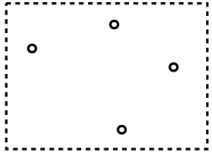
Pixels

Pixel Operations

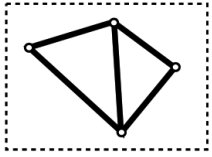
ทางออกท่อ ออก display

1 2 3 4 Vertices in 3D space

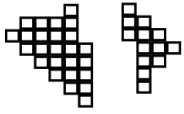
Code เขียนได้ 2 ส่วนหลัก
1) vertex program
2) fragments (กำหนดสี)



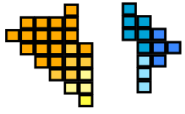
Vertices in positioned on screen



Triangles positioned on screen



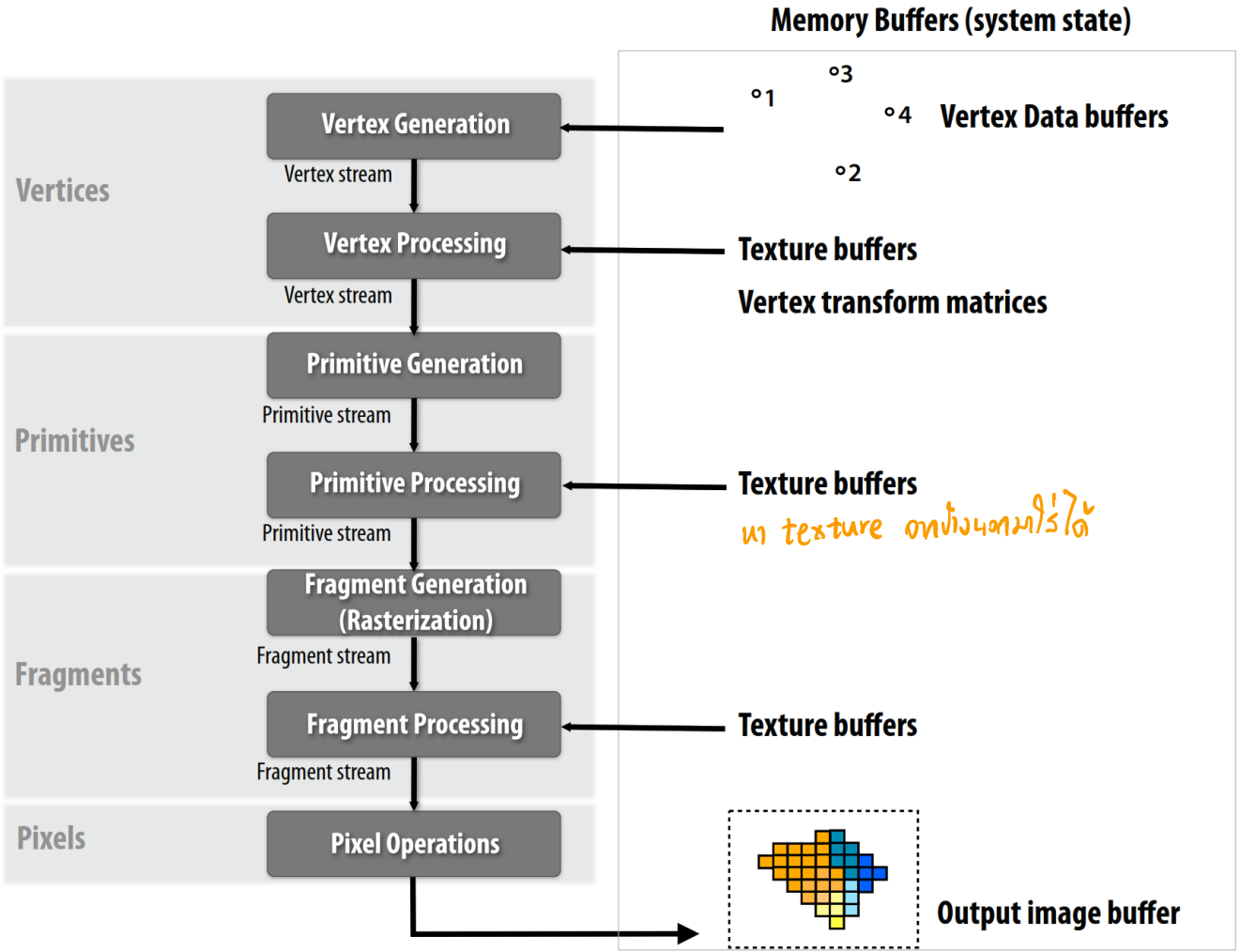
Fragments (one per each covered pixel)



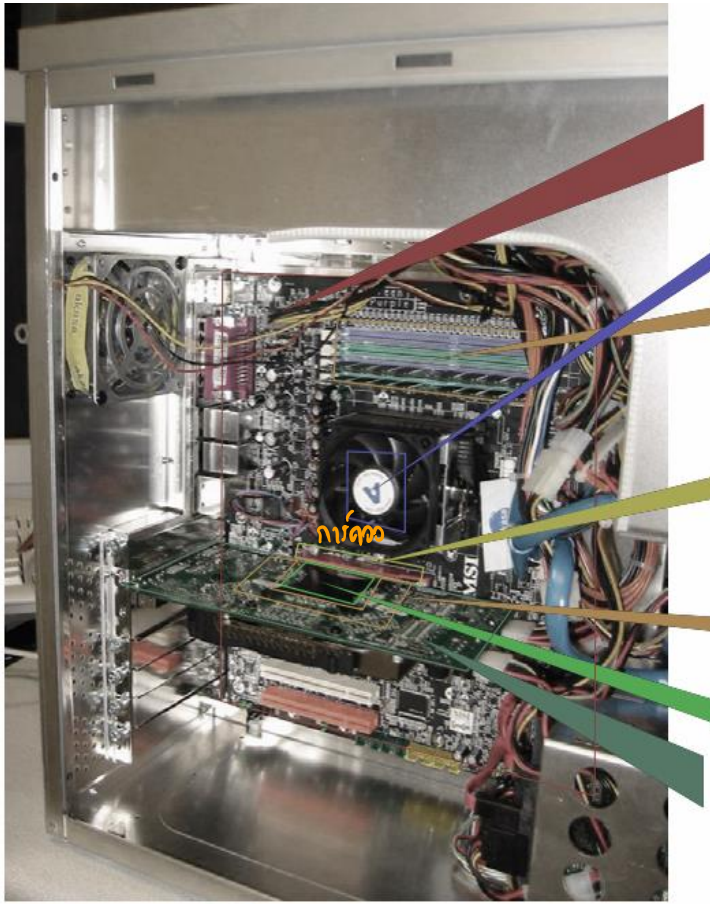
Shaded fragments
คำนวณ shaded ให้ทุกๆ fragments

Output image (pixels)

Real-time graphics pipeline (state)



PC Architecture



Motherboard

Central Processor Unit (CPU)

System Memory

Bus Port (PCI, AGP, PCIe)

Video Memory

Graphics Processor Unit (GPU)

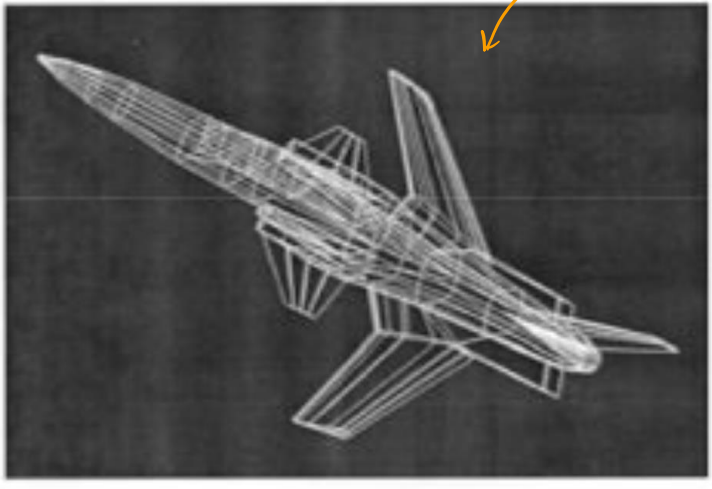
Video Board

“Historic” Phase
Evolution of Interactive 3D Graphics

First generation - wireframe เส้นเป็นหลัก

- Vertex: transform, clip, and project
- Rasterization: color interpolation (points, lines)
- Fragment: overwrite
- Dates: prior to 1987

ลักษณะเป็นเส้น ไม่มี shade



Slide credit Kurt Akeley and Pat Hanrahan

Second generation - shaded solids

Vertex: lighting calculation

⇨ Rasterization: depth interpolation (triangles)

⇨ Fragment: depth buffer, color blending

Dates: 1987 - 1992

Slide credit Kurt Akeley and Pat Hanrahan

Third generation - texture mapping

Vertex: texture coordinate transformation

⇨ Rasterization: texture coordinate interpolation

⇨ Fragment: texture evaluation, antialiasing

Dates: 1992 - 2000

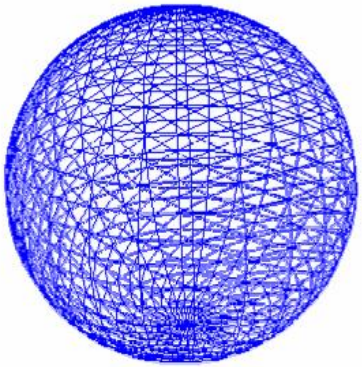
เอารูปภาพมาแปะ

เอารูปภาพมา map

ทุกวันนี้ภาพทำ texture mapping ยังทำอยู่

Texture Mapping

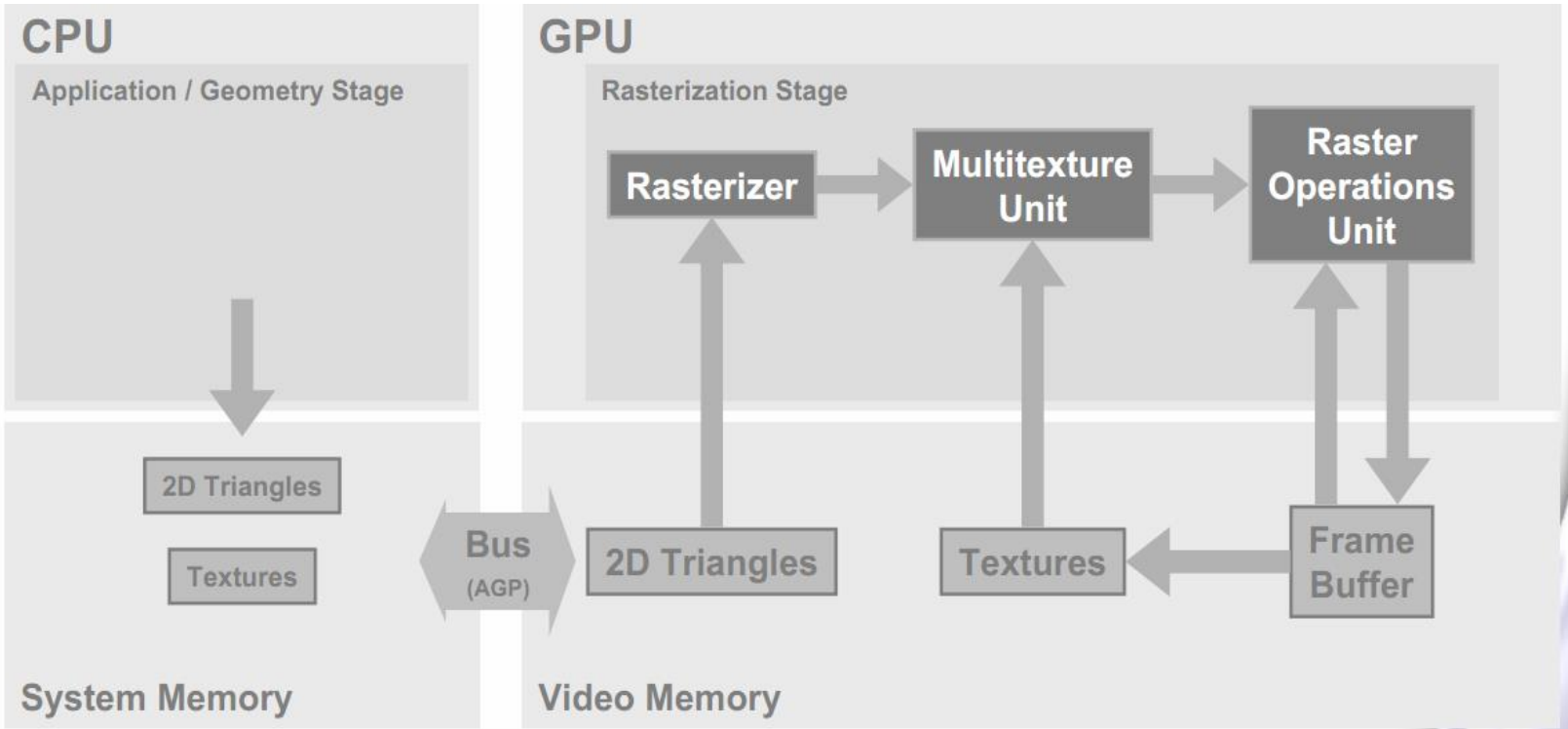
Triangle Mesh textured with Base Texture



+

=

1998: Multitexturing



- AGP: Accelerated Graphics Port
- NVIDIA's TNT, ATI's Rage

Multitexturing

Base Texture modulated by Light Map

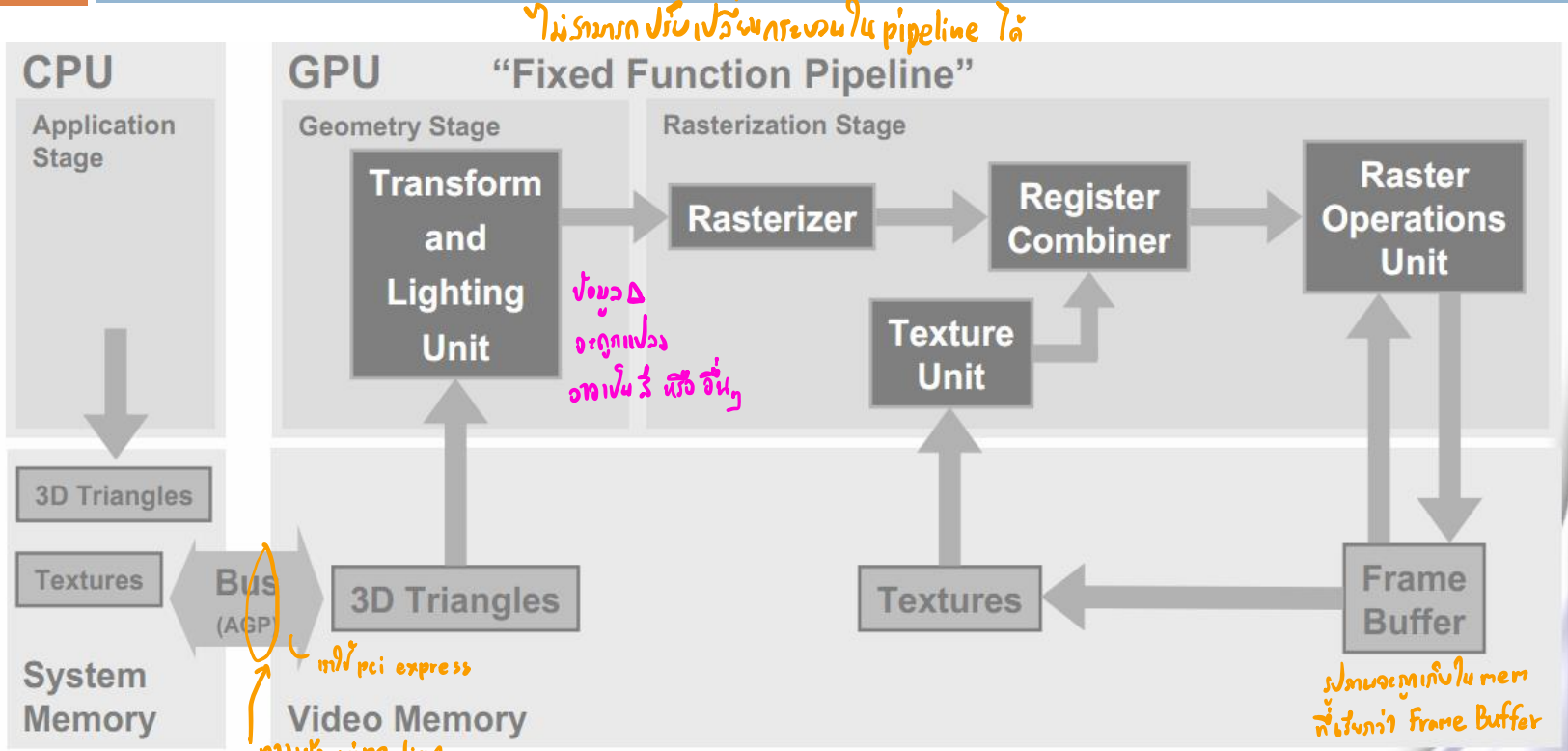
blend

X

=

from UT2004 (c)
Epic Games Inc.
Used with permission

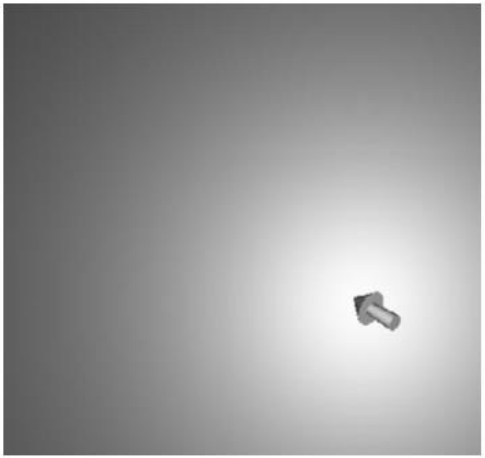
1999-2000: Transform and Lighting



- **Register Combiner**: Offers many more texture/color combinations
- NVIDIA's GeForce 256 and GeForce2, ATI's Radeon 7500, S3's Savage3D

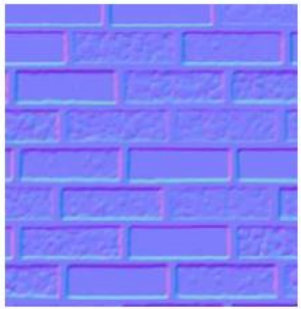
Bump Mapping เร็ว

- Bump mapping is about fetching the normal from a texture (called a normal map) instead of using the interpolated normal to compute lighting at a given pixel



Diffuse light without bump

+ bump map



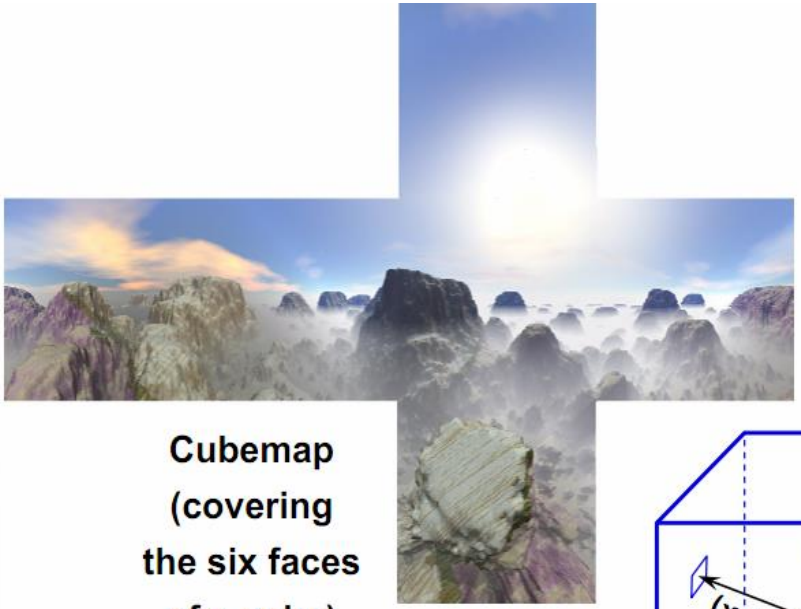
Normal Map

=

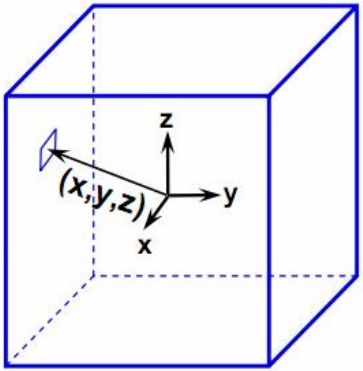
Diffuse light with bumps

เป็นเทคนิคที่เร็วมาก

Cube Texture Mapping



Cubemap
(covering
the six faces
of a cube)



Cubemap lookup
(with direction (x, y, z))

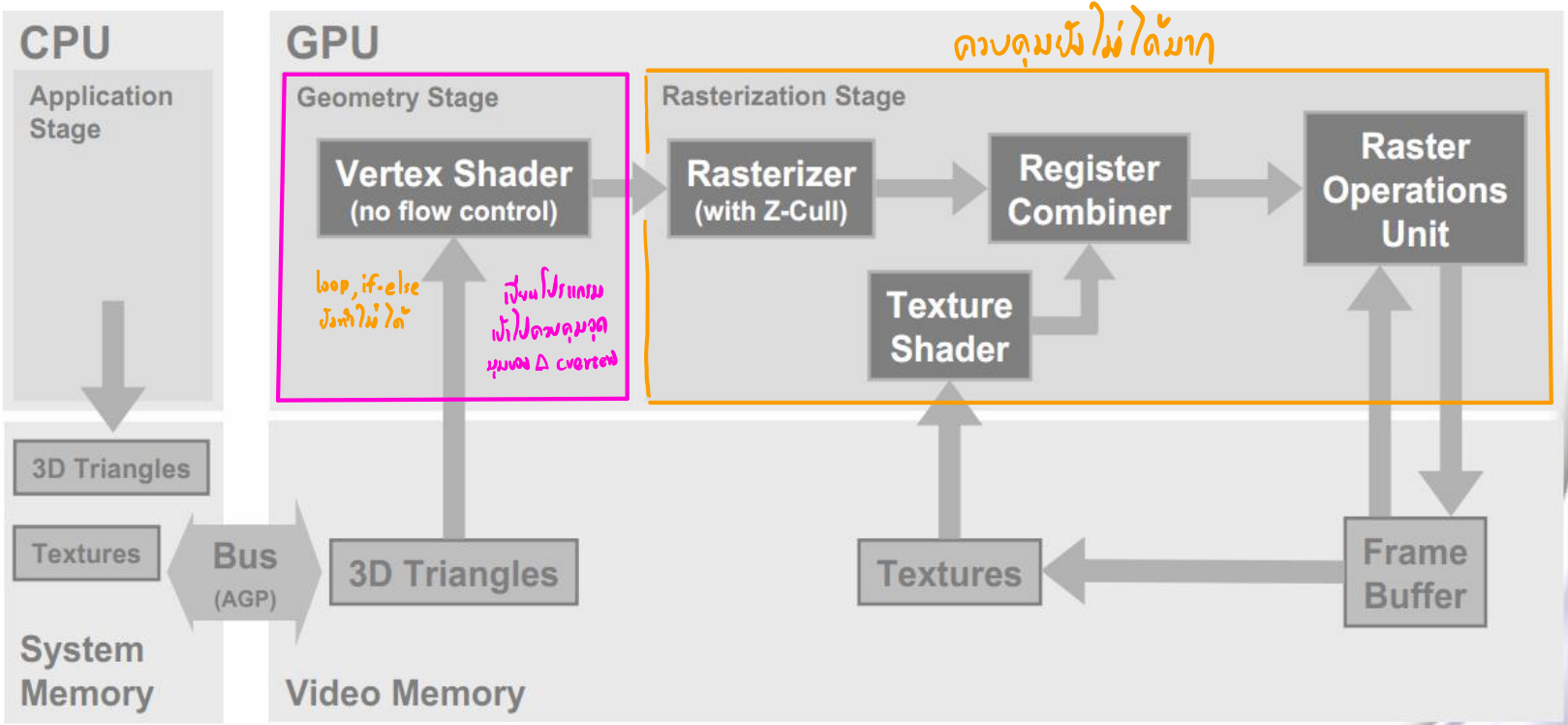
Environment Mapping
(the reflection vector
is used to lookup
the cubemap)

NVIDIA.

Programable เขียนโปรแกรม เข้าไปควบคุมการทำงาน ของการ์ดจอ หรือ การทำงาน ในส่วนต่างของ pipe line ในการ์ดจอ

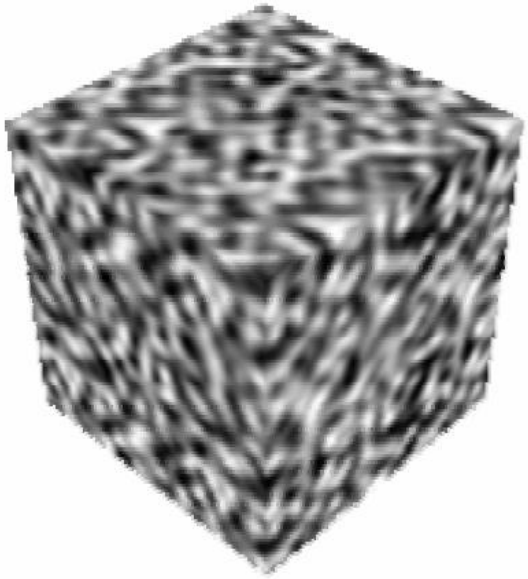
“Modern” Phase
Evolution of Interactive 3D Graphics

2001: Programmable Vertex Shader

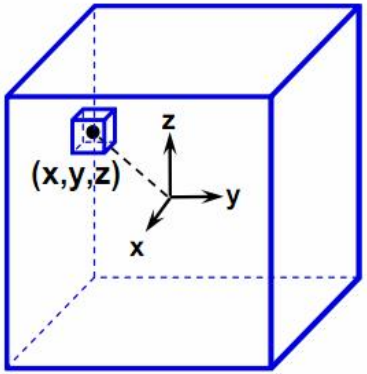


- **Z-Cull**: Predicts which fragments will fail the Z test and discards them
- **Texture Shader**: Offers more texture addressing and operations
- **NVIDIA's GeForce3 and GeForce4 Ti, ATI's Radeon 8500**

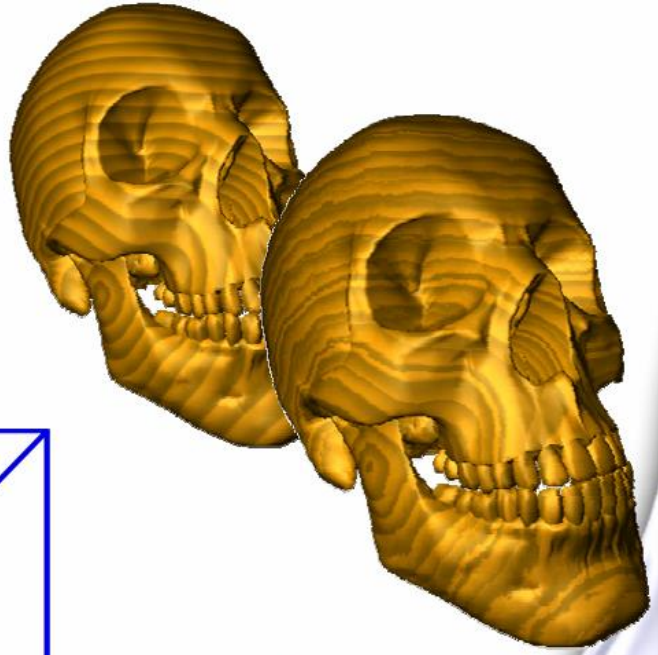
Volume Texture Mapping



Volume Texture
(3D Noise)



Volume Texture lookup
(with position (x, y, z))

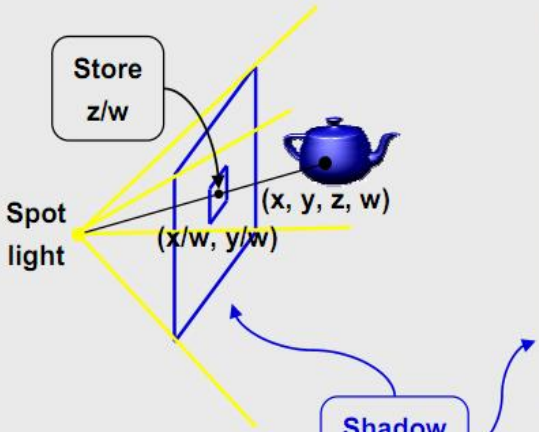


Noise Perturbation

Hardware Shadow Mapping

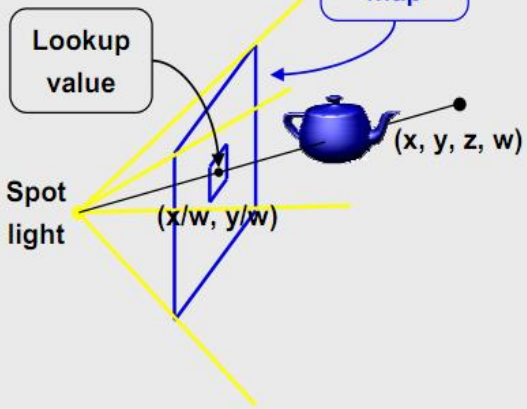
Shadow Map Computation

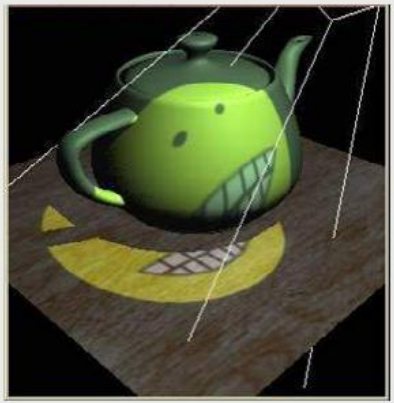
The shadow map contains
the depth z/w of the 3D points visible
from the light's point of view



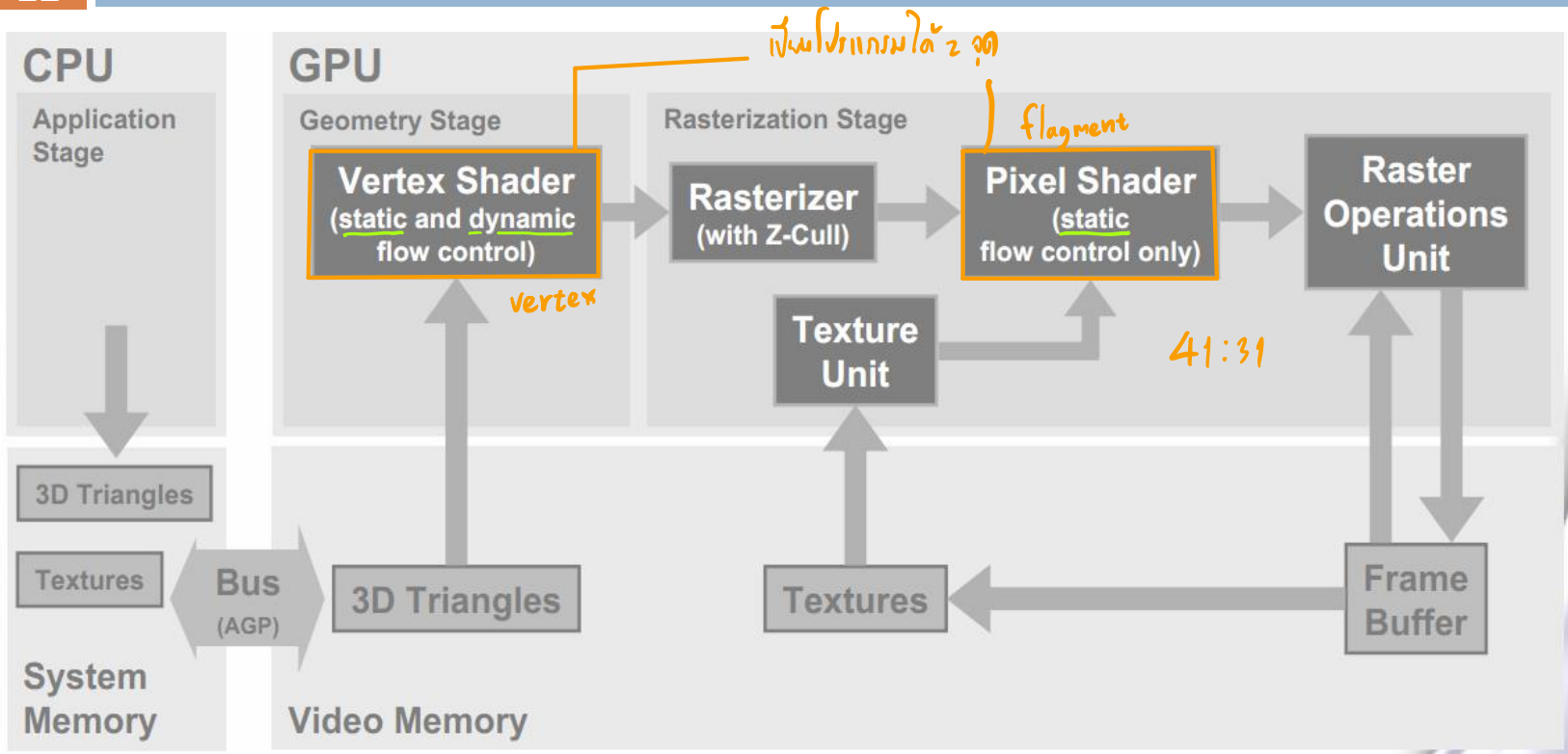
Shadow Rendering

A 3D point (x, y, z, w) is in shadow if:
z/w < value of shadow map at (x/w, y/w)
A hardware shadow map lookup
returns the value of this comparison
between 0 and 1





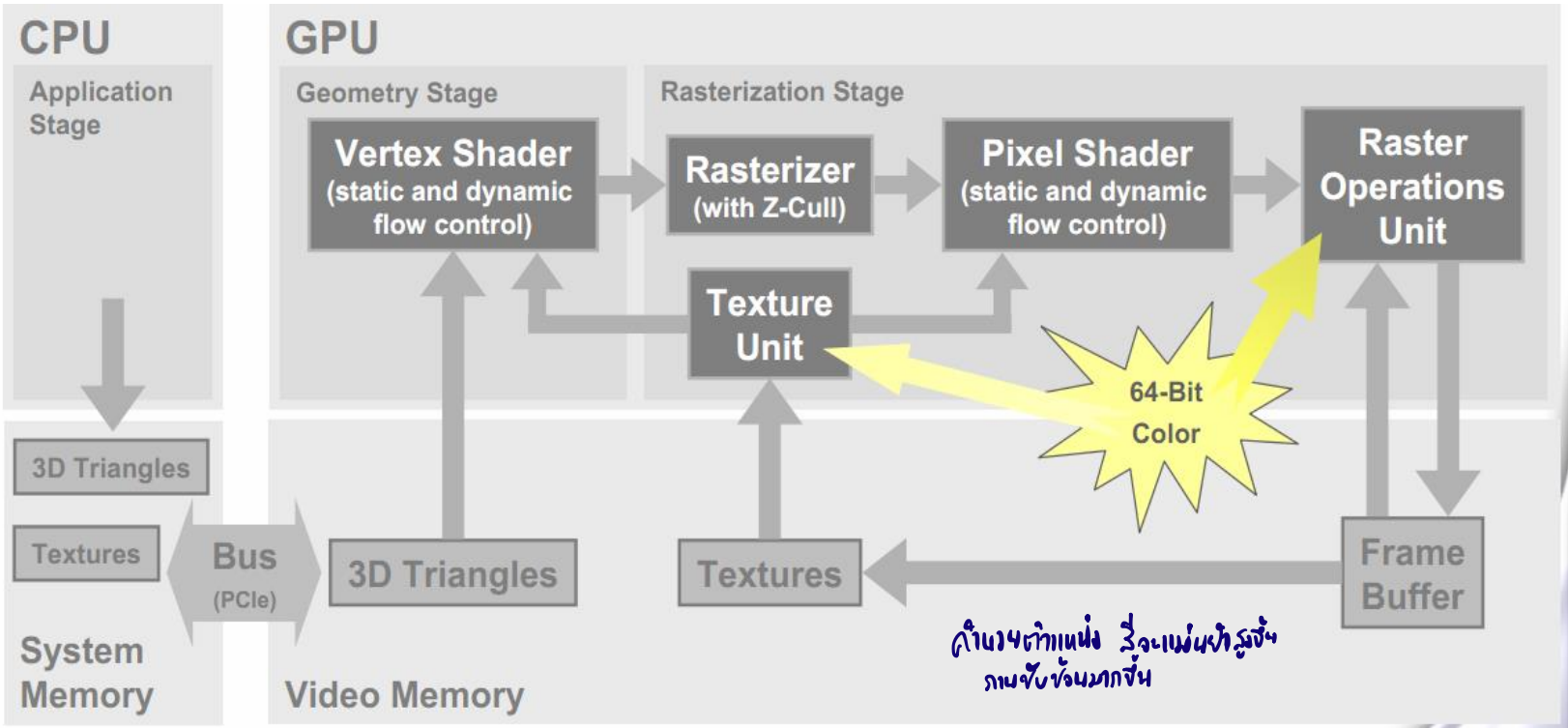
2002-03: Programmable Pixel Shader



- MRT: Multiple Render Target
- NVIDIA's GeForce FX, ATI's Radeon 9600 to 9800 and X600 to X800

2004: Shader Model 3 & 64-Bit Color

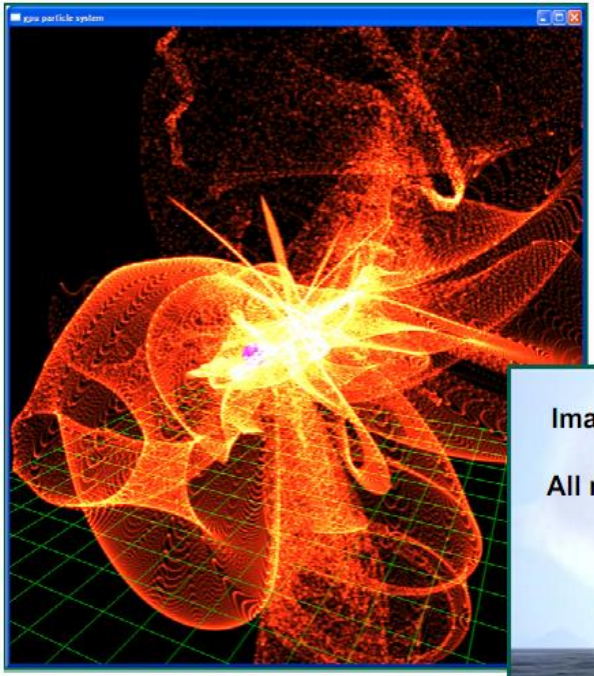
เก็บข้อมูลได้ละเอียดขึ้น (double presition)

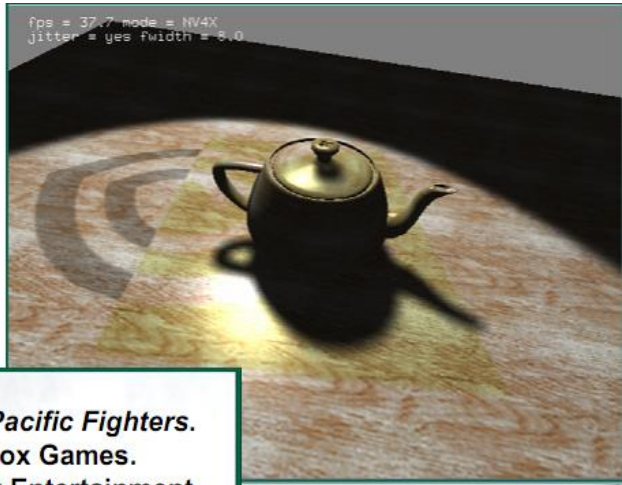


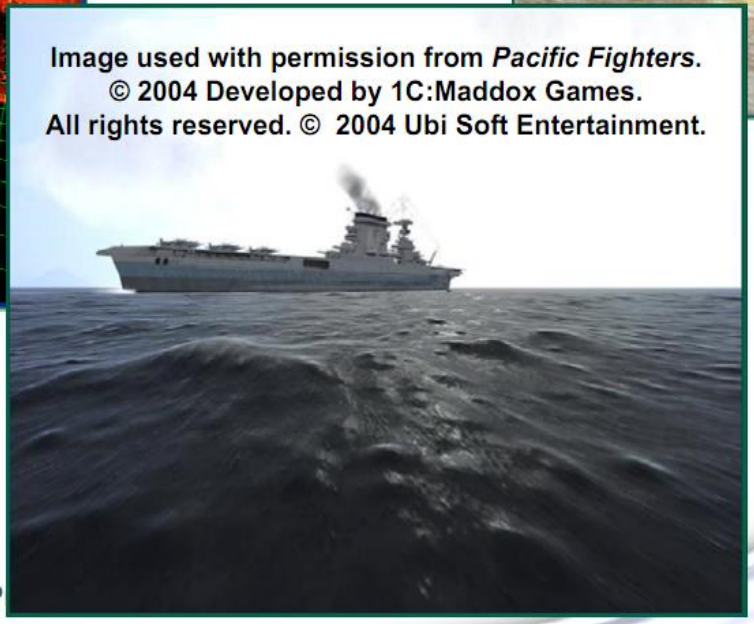
คำนวณตำแหน่ง สีจะแม่นยำขึ้น ภาพชัดขึ้นมากขึ้น

- PCIe: Peripheral Component Interconnect Express
- NVIDIA's GeForce 6 Series (6800 and 6600)

Shader Model 3.0 Unleashed







NVIDIA.

64-Bit Color Support

- 64-bit color means one 16-bit floating-point value per channel (R, G, B, A)
- Alpha blending works with 64-bit color buffer
  (as opposed to 32-bit fixed-point color buffer only)
- Texture filtering works with 64-bit textures
  (as opposed to 32-bit fixed-point textures only)
- Applications:
  - High-precision image compositing
  - High dynamic range imagery (HDR)

High Dynamic Range Imagery

- The dynamic range of a scene is the ratio of the highest to the lowest luminance
- Real-life scenes can have high dynamic ranges of several millions
- Display and print devices have a low dynamic range of around 100
- Tone mapping is the process of displaying high dynamic range images on those low dynamic range devices
- High dynamic range images use floating-point colors
- OpenEXR is a high dynamic range image format that is compatible with NVIDIA's 64-bit color format

Real-Time Tone Mapping

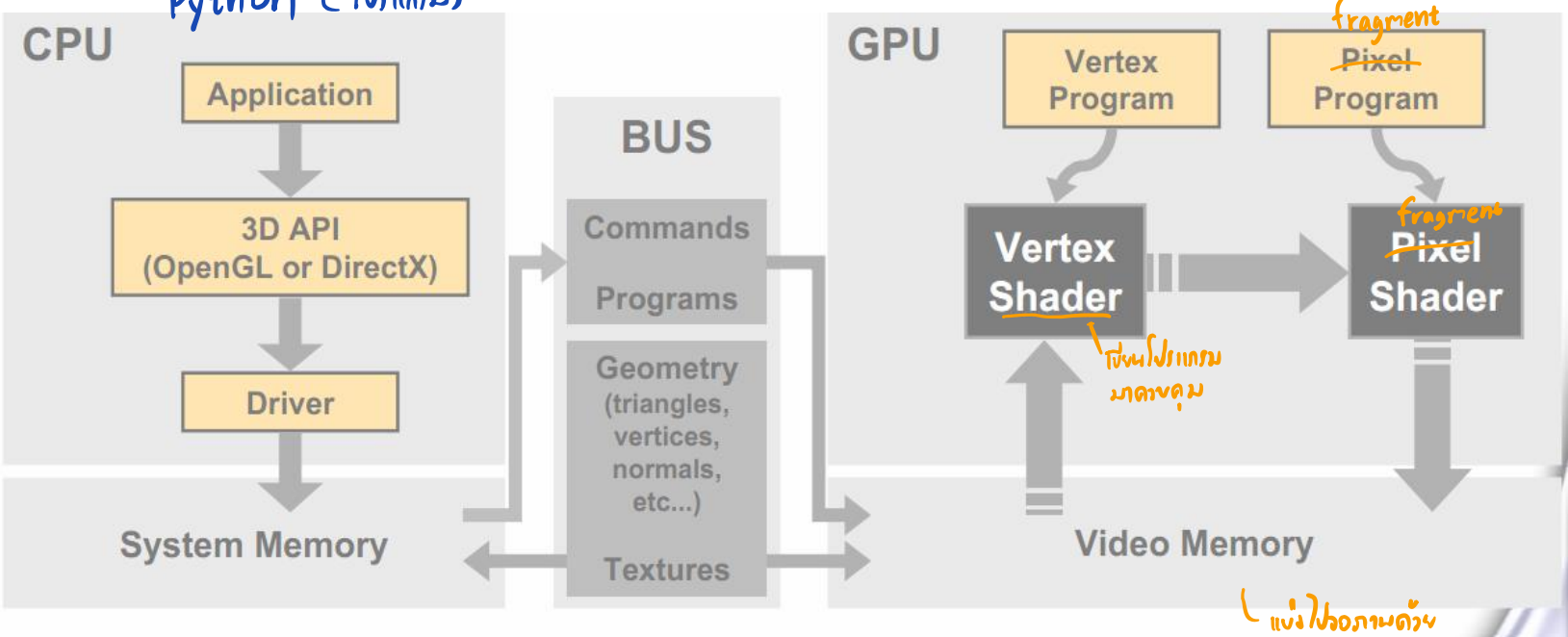
- The image is entirely computed in 64-bit color and tone-mapped for display

From low to high exposure image of the same scene

PC Graphics Software Architecture

PC Graphics Software Architecture

Python (โปรแกรม)



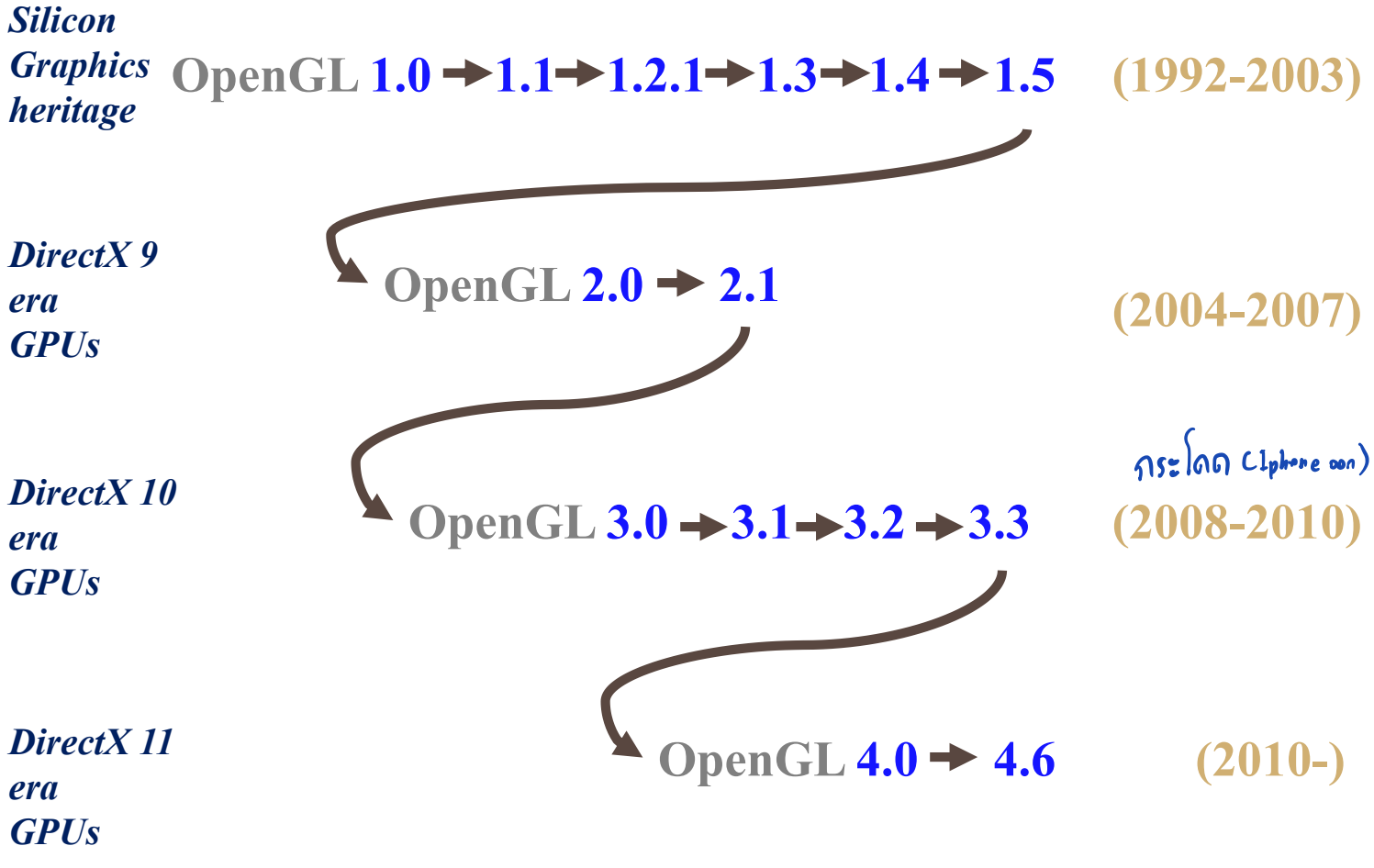
โปรแกรมควบคุม

(แบ่งใช้ของภาพด้วย

- The application, 3D API and driver are written in C or C++
- The vertex and pixel programs are written in a high-level shading language (Cg, DirectX HLSL, OpenGL Shading Language)

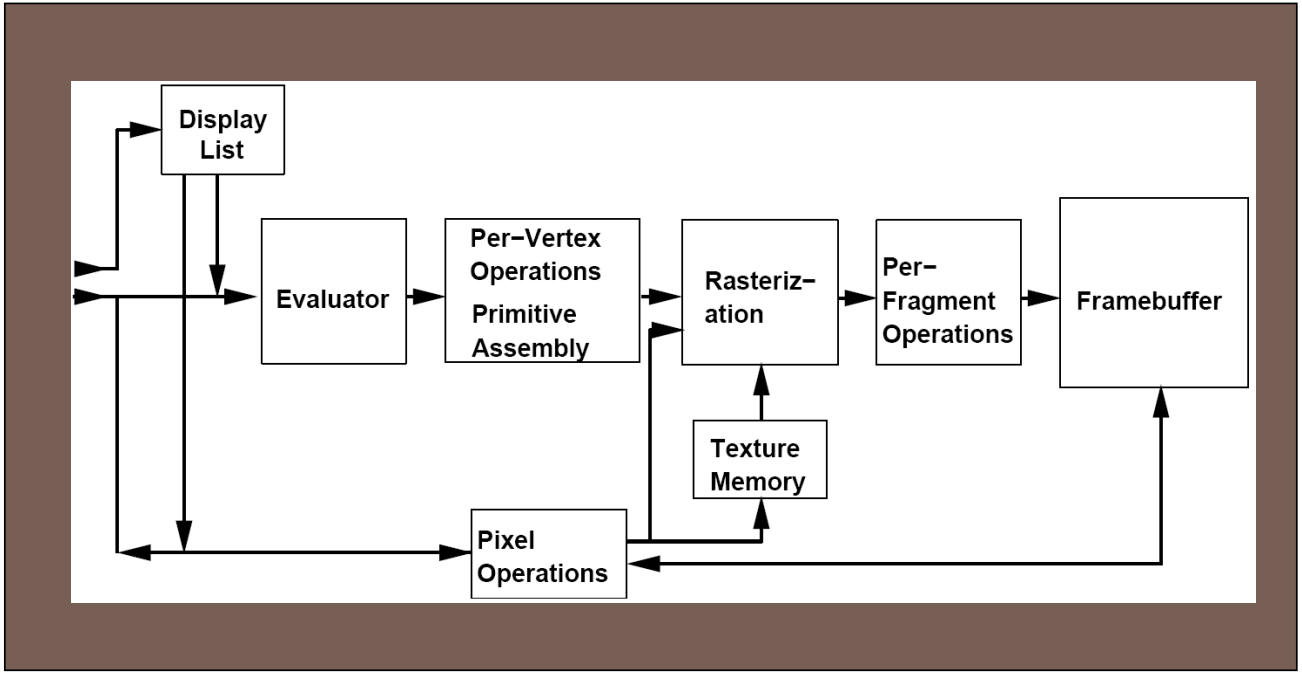
1. Nvidia เมื่อก่อน micro soft

OpenGL Version Progression

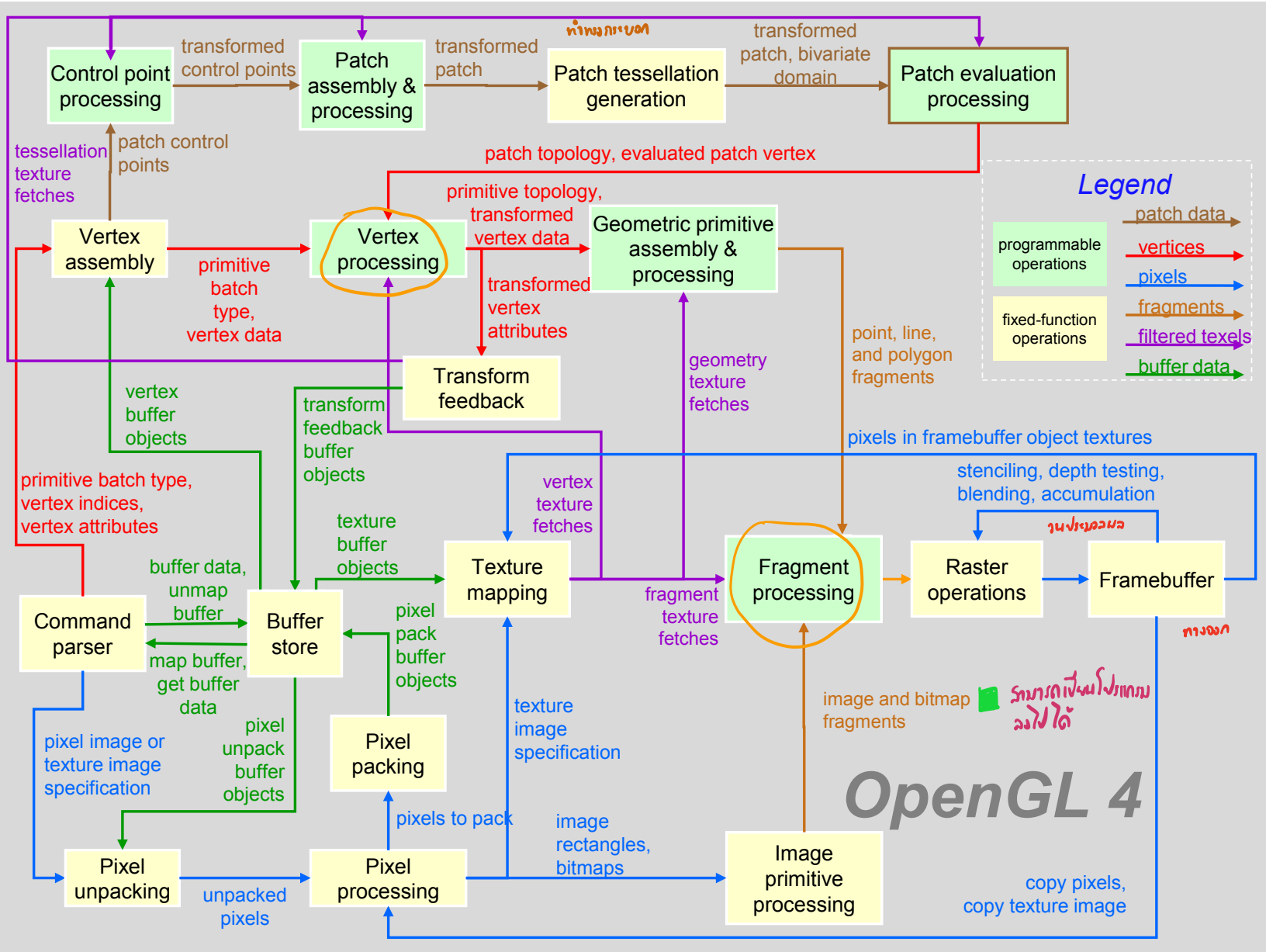


Classic OpenGL State Machine

- From 1991-2007
  - vertex & fragment processing became programmable



[source: GL 1.0 specification]


Control point processing
transformed control points
Patch assembly & processing
transformed patch
Patch tessellation generation
transformed patch, bivariate domain
Patch evaluation processing
tessellation texture fetches
patch control points
patch topology, evaluated patch vertex
Legend
patch data
vertices
pixels
fragments
filtered texels
buffer data
programmable operations
fixed-function operations
Vertex assembly
primitive batch type, vertex data
Vertex processing
primitive topology, transformed vertex data
Geometric primitive assembly & processing
transformed vertex attributes
point, line, and polygon fragments
Transform feedback
geometry texture fetches
vertex buffer objects
transform feedback buffer objects
pixels in framebuffer object textures
primitive batch type, vertex indices, vertex attributes
vertex texture fetches
stenciling, depth testing, blending, accumulation
texture buffer objects
buffer data, unmap buffer
Texture mapping
Fragment processing
Raster operations
Framebuffer
Command parser
Buffer store
map buffer, get buffer data
pixel pack buffer objects
fragment texture fetches
image and bitmap fragments
pixel image or texture image specification
pixel unpack buffer objects
Pixel packing
texture image specification
pixels to pack
image rectangles, bitmaps
OpenGL 4
Pixel unpacking
unpacked pixels
Pixel processing
Image primitive processing
copy pixels, copy texture image

OpenGL 4.0 Programmable
Features: Iso-line Tessellation Shaders

- Intended for hair, particular animation and combing
  - Scientific visualization applications too

References

- Fatahalian, K. (2011). CMU 15-869 Graphics and Imaging Architectures [Lecture notes]. Retrieved from www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15869-f11/www.
- Kilgard, M. and Lichtenbelt, B. 2010. *OpenGL 4 for 2010.* Presented at Siggraph 2010, Los Angeles, July 28, 2010.
- Zeller, C. 2004. *Introduction to the Hardware Graphics Pipeline.* Tutorial presented at Eurographics 2004, Grenoble (France). Retrieved from developer.nvidia.com/system/files/akamai/gamedev/docs/EG_04_IntroductionToGPU.pdf, [Dec. 12, 2011] .