

Evolution of GPUs

Chris Seitz



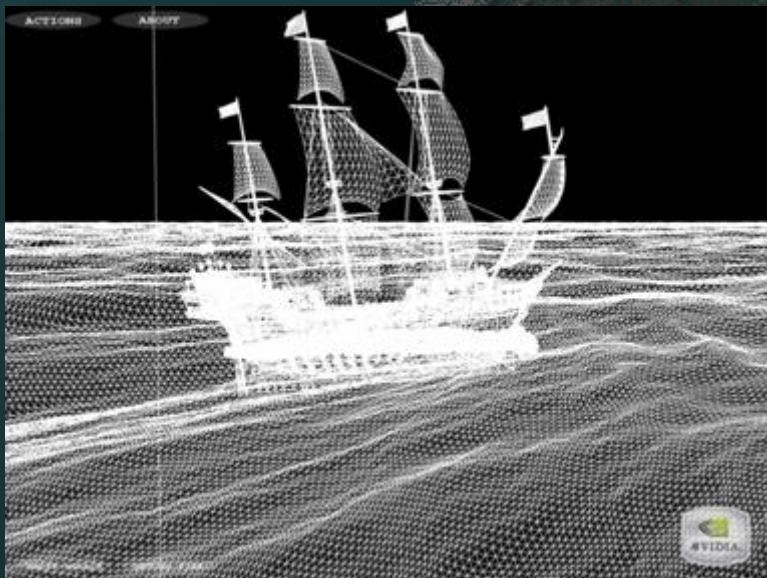
Overview

- Concepts:
 - Real-time rendering
 - Hardware graphics pipeline
- Evolution of the PC hardware graphics pipeline:
 - 1995-1998: Texture mapping and z-buffer
 - 1998: Multitexturing
 - 1999-2000: Transform and lighting
 - 2001: Programmable vertex shader
 - 2002-2003: Programmable pixel shader
 - 2004: Shader model 3.0 and 64-bit color support

Real-Time Rendering



- Graphics hardware enables real-time rendering
- Real-time means display rate at more than 10 images per second

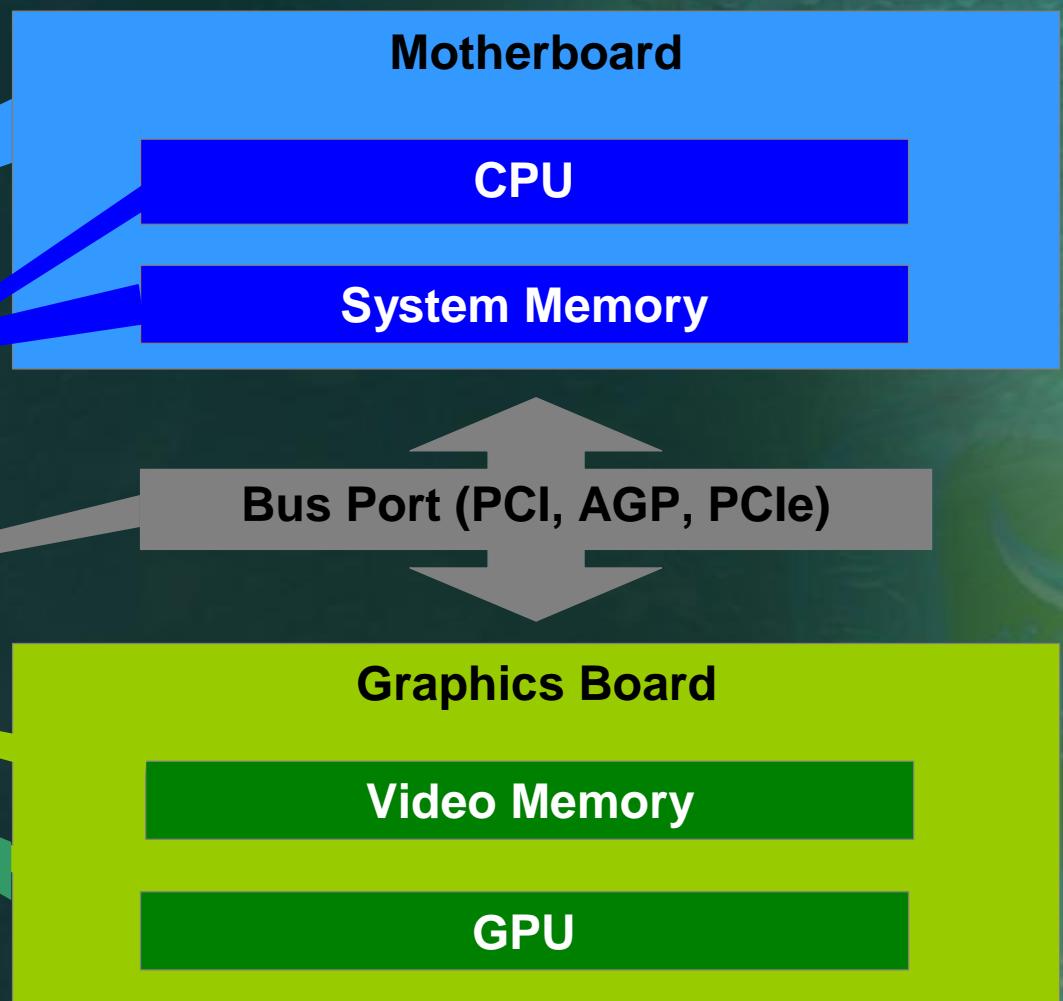
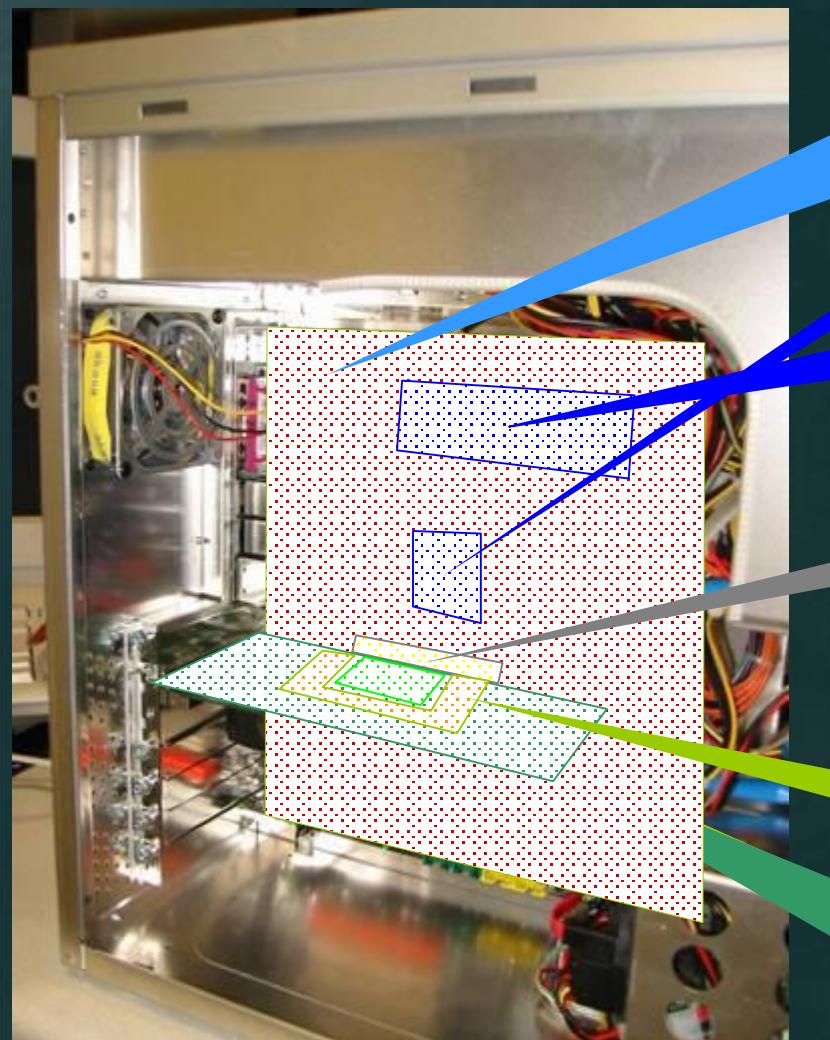


3D Scene =
Collection of
3D primitives (triangles, lines, points)



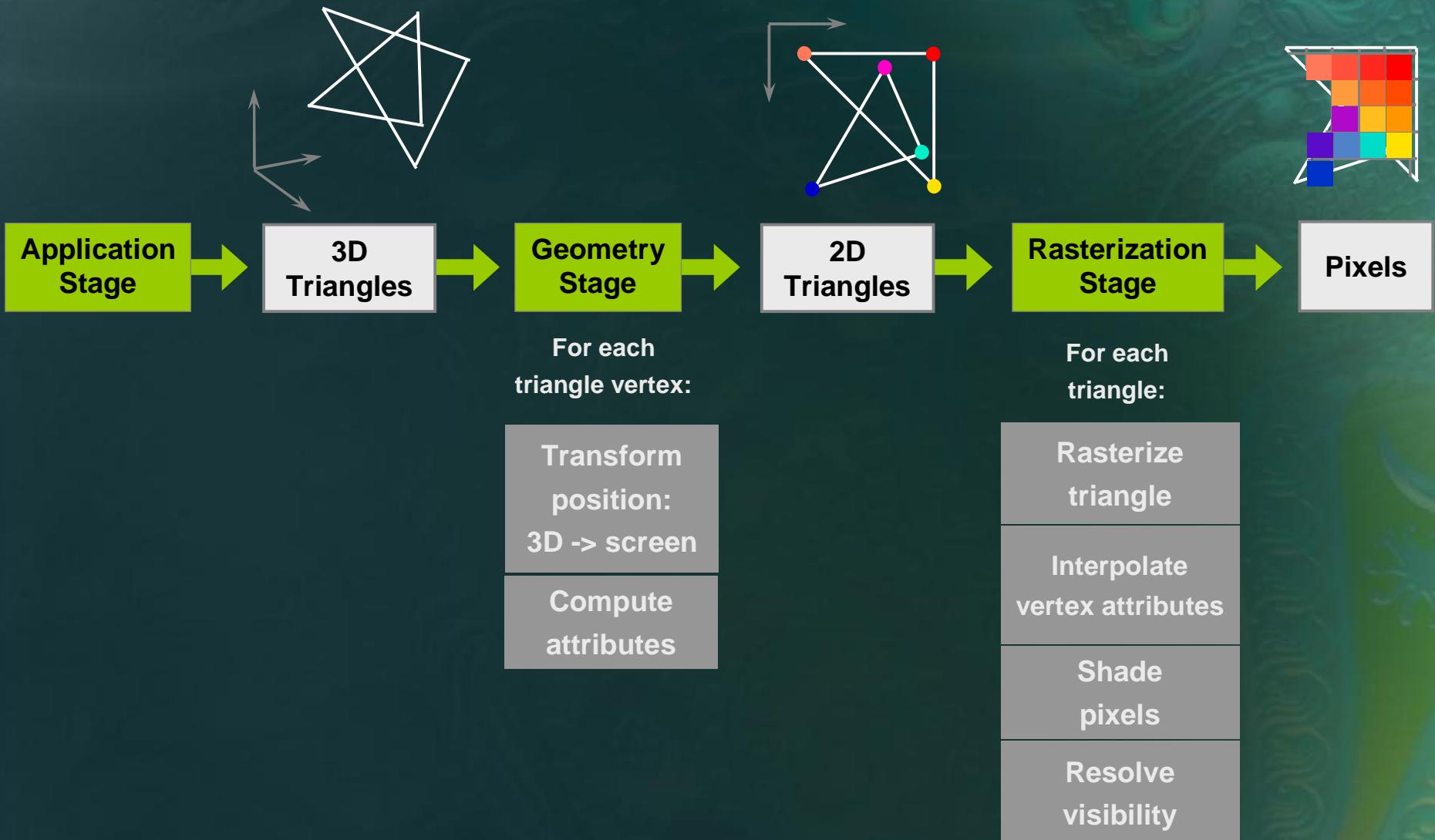
Image =
Array of pixels

PC Architecture



©2004 NVIDIA Corporation. All rights reserved.

Hardware Graphics Pipeline



Real-time Graphics 1997: RIVA 128 – 3M Transistors



©2004 NVIDIA Corporation. All rights reserved.

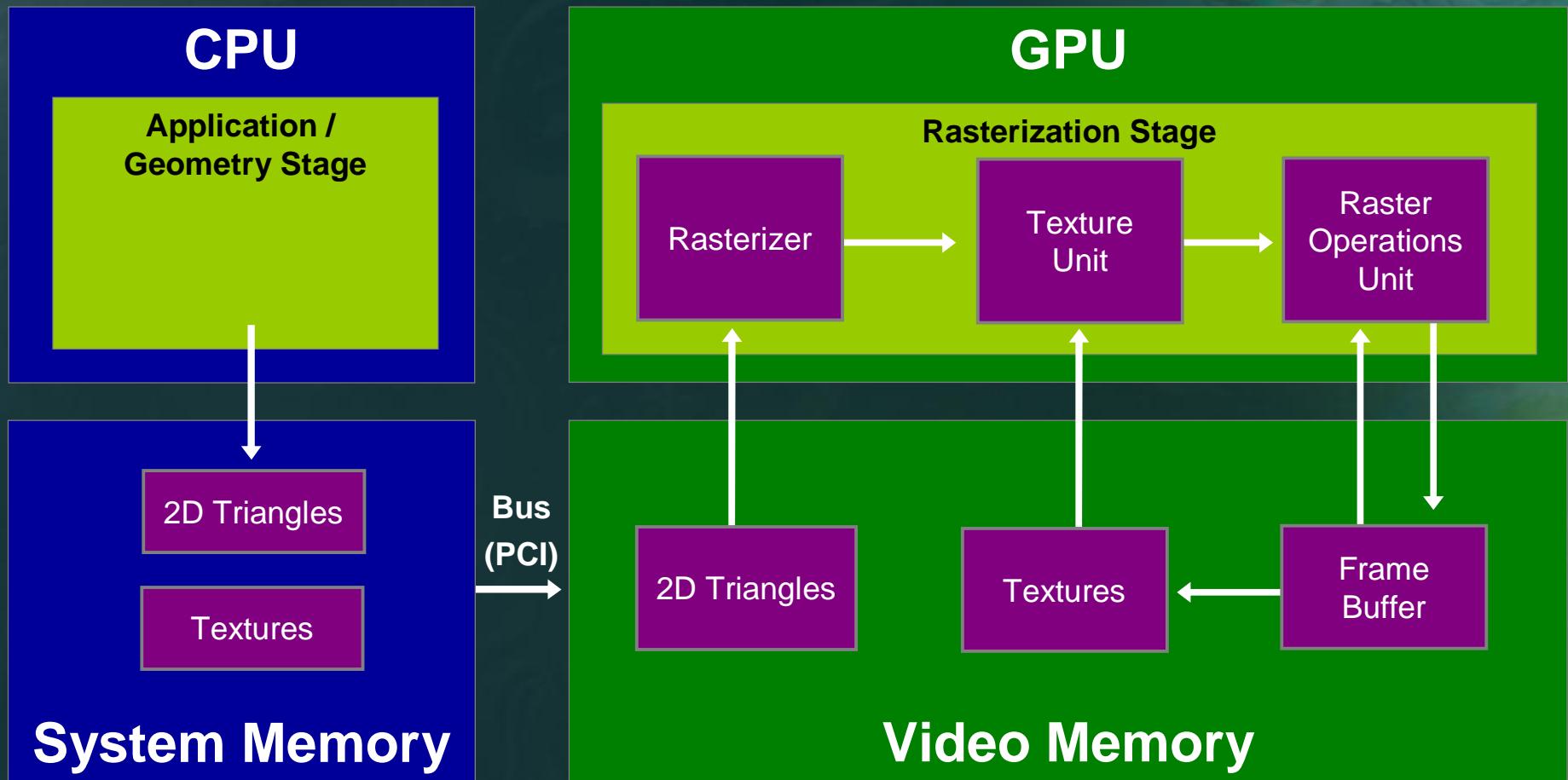
Real-time Graphics 2004:



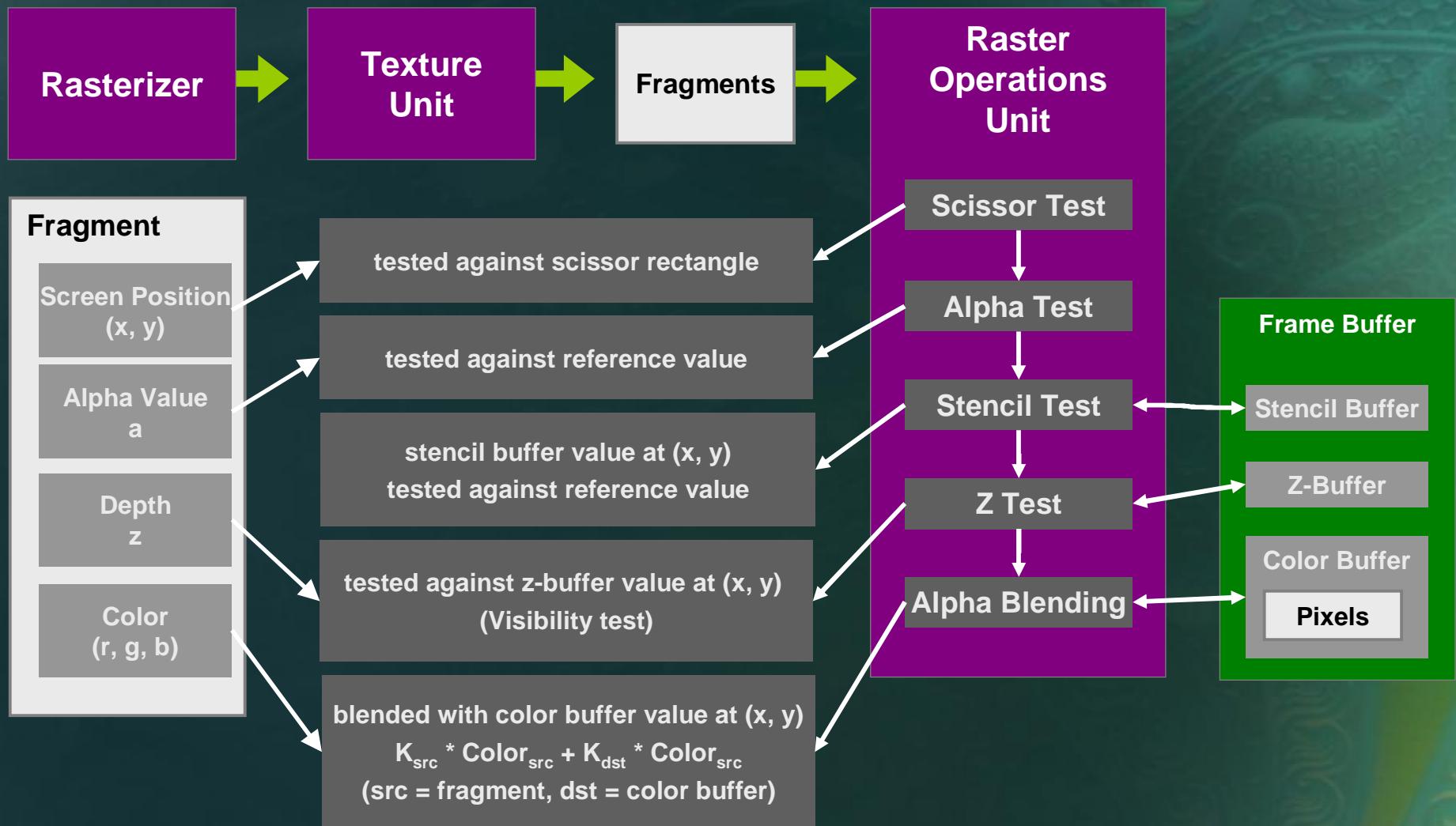
© UnReal Engine 3.0 Images Courtesy of Epic Games

Unreal Engine 3 Technology Demo © 2004 Epic Games. All Rights Reserved

'95-'98: Texture Mapping & Z-Buffer



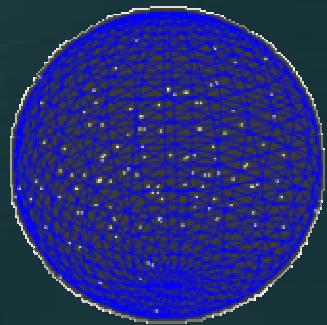
Raster Operations Unit (ROP)



Texture Mapping



Triangle Mesh
(with UV coordinates)



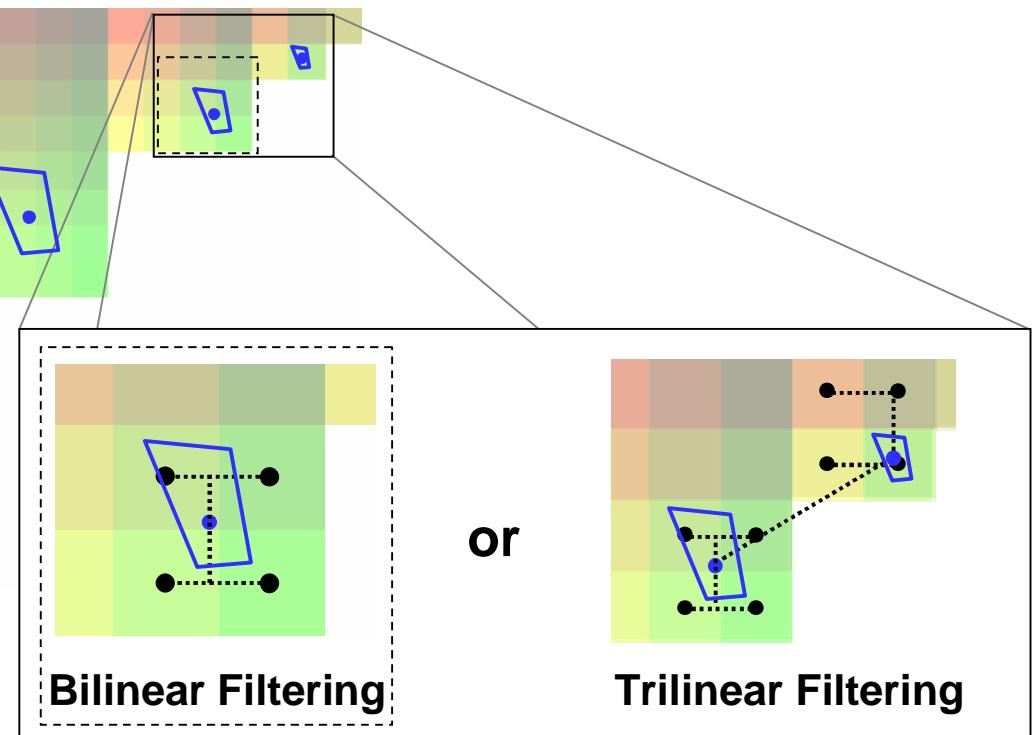
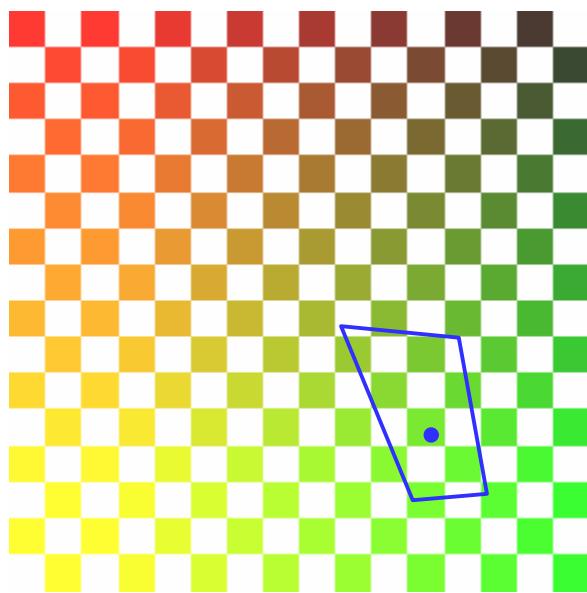
Base Texture



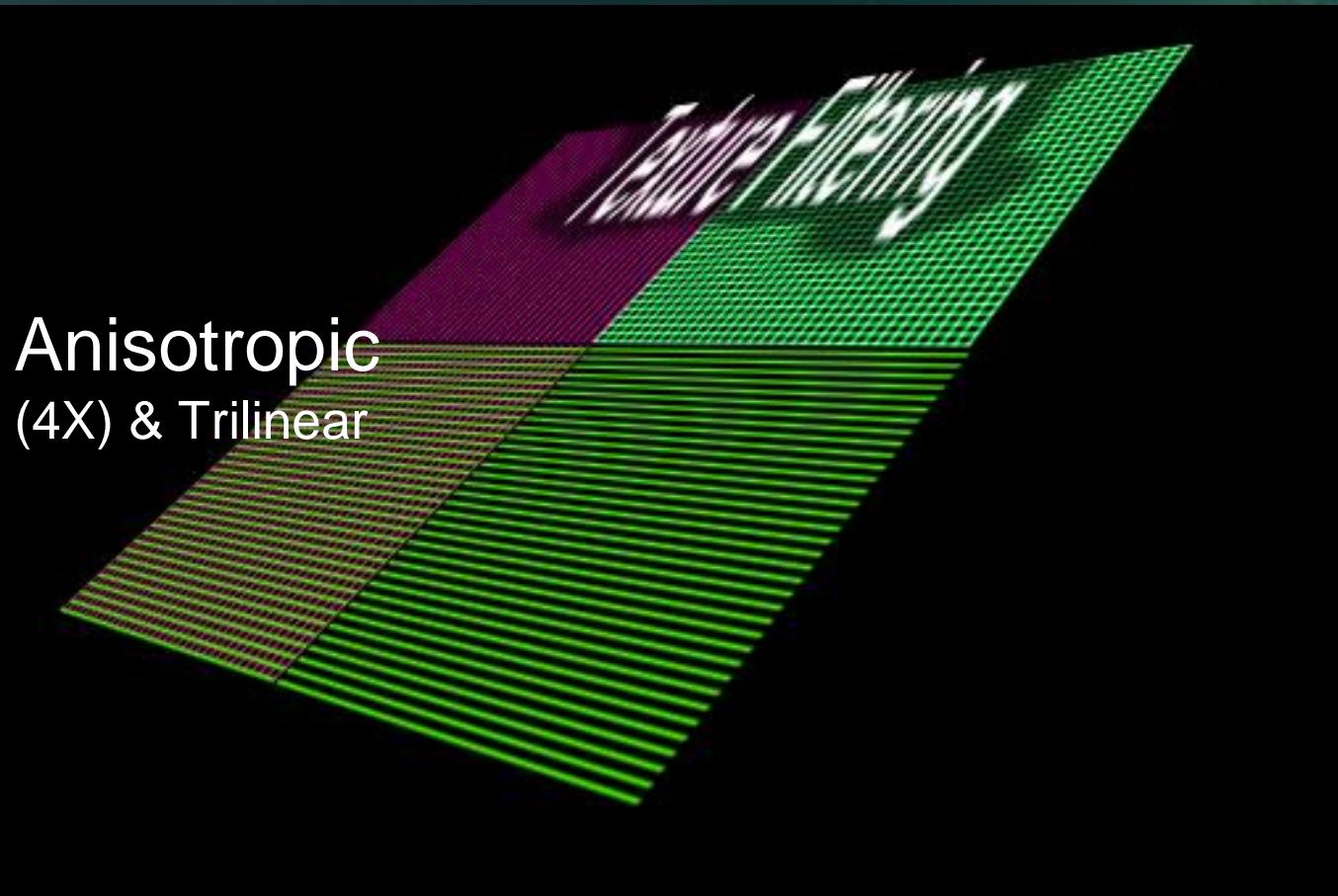
- Sampling
 - Magnification, Minification
- Filtering
 - Bilinear, Trilinear, Anisotropic
- Mipmapping
- Perspective Correct Interpolation

©2004 NVIDIA Corporation. All rights reserved.

Texture Mapping: Mipmapping & Filtering



Filtering Examples



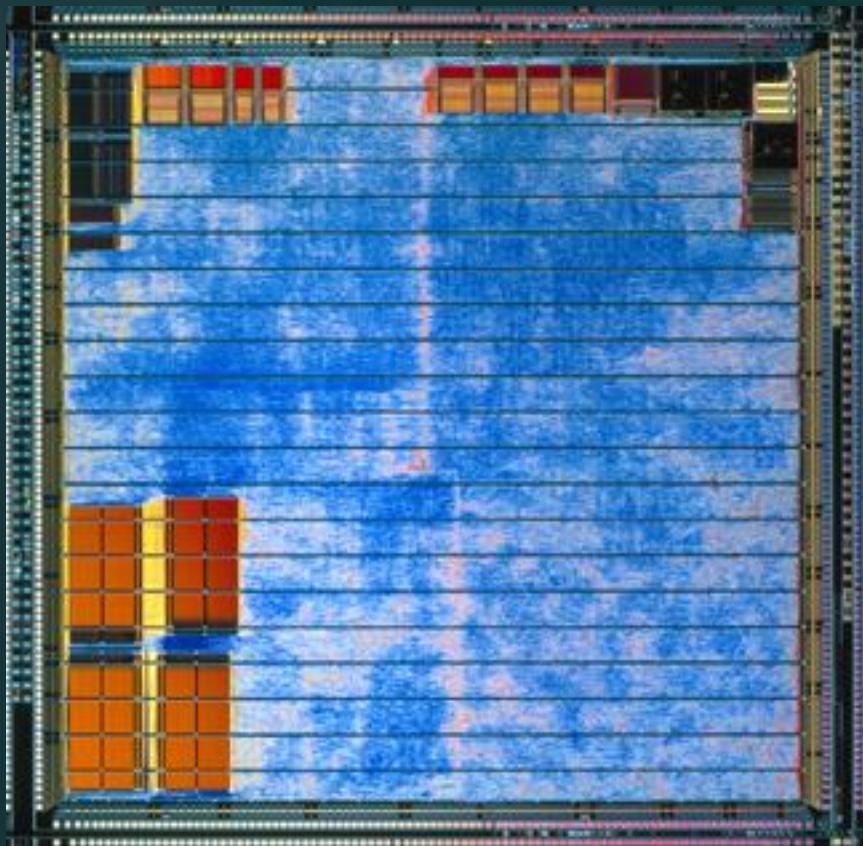
©2004 NVIDIA Corporation. All rights reserved.

Incoming



©2004 NVIDIA Corporation. All rights reserved.

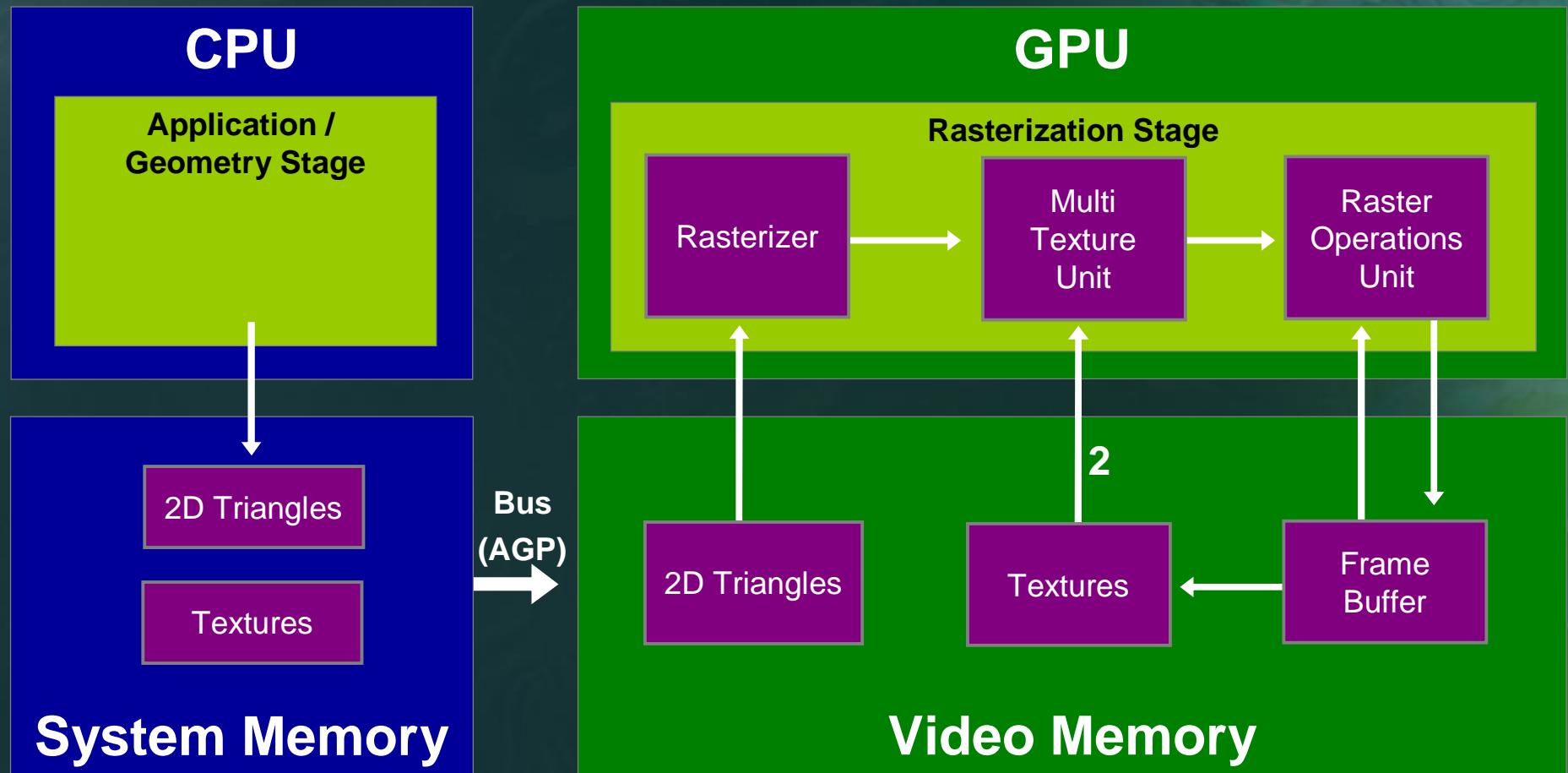
Riva TNT – 7M Transistors - 1998



Multitexture



1998: Multitexturing



Multitexturing



Base Texture



modulated by

X



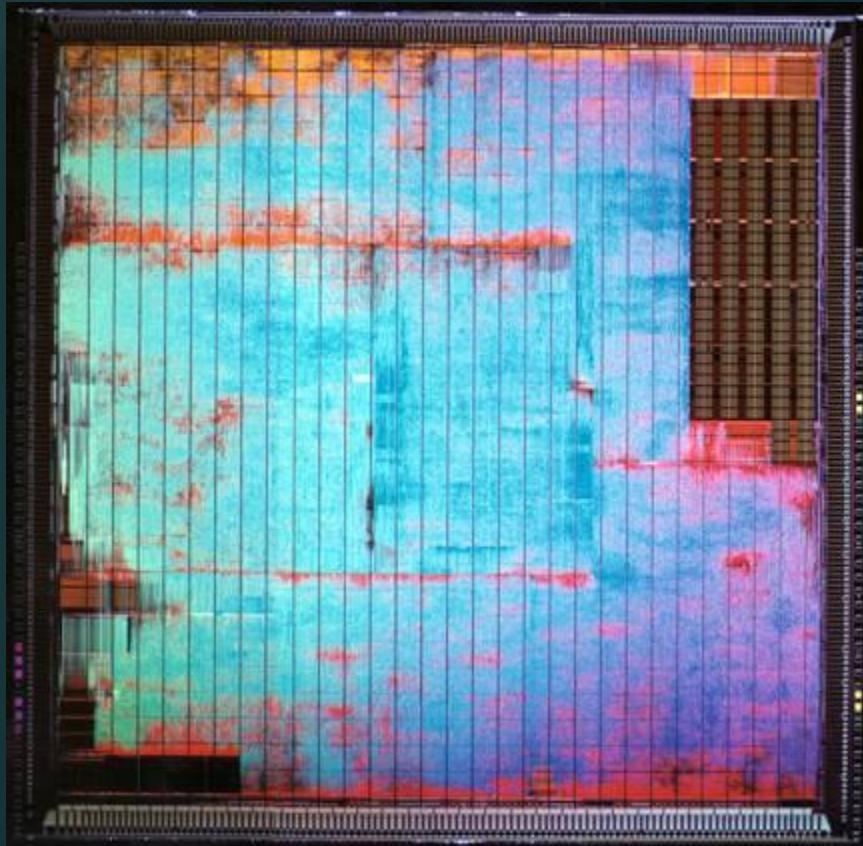
Light Map

=



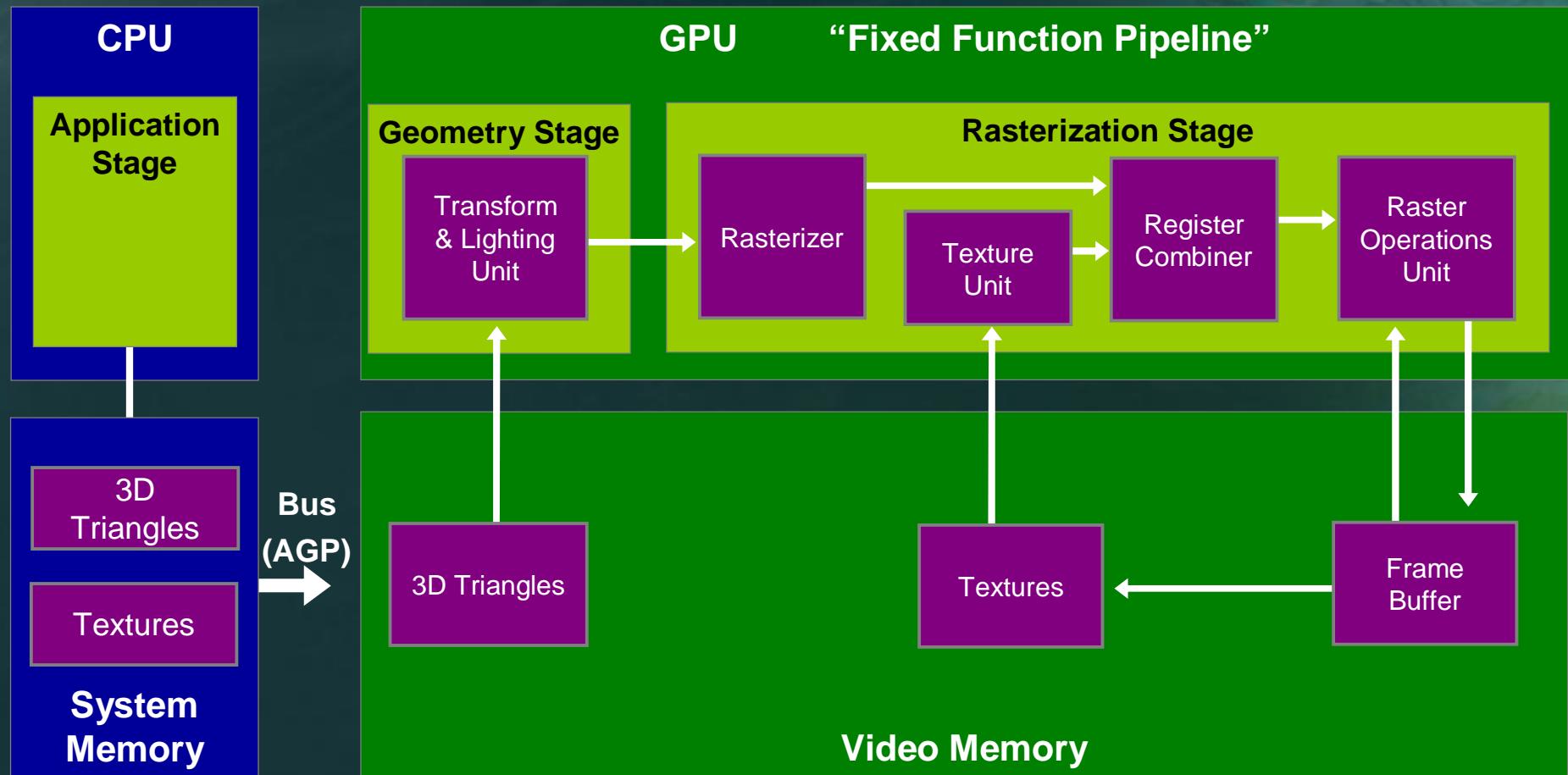
from UT2004 (c)
Epic Games Inc.
Used with permission

GeForce 256 – 23M Transistors - 1999



**Hardware
Transform &
Lighting**

1999-2000: Transform and Lighting

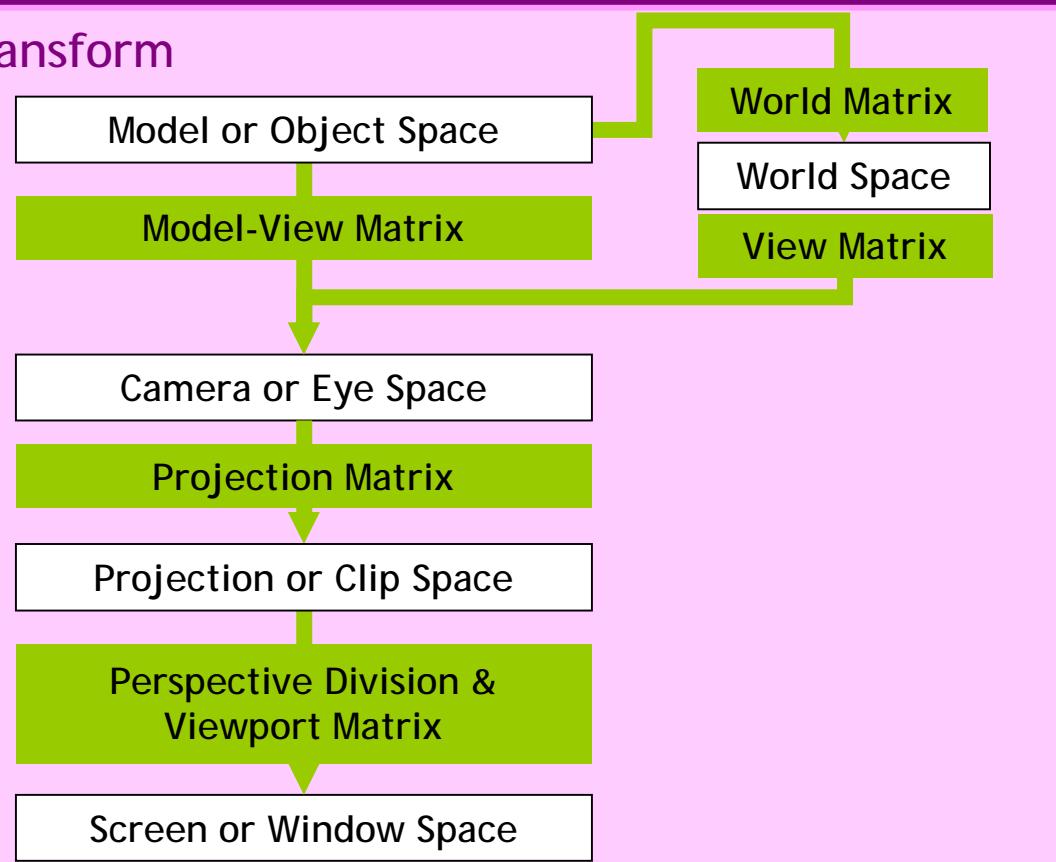


Transform and Lighting Unit (TnL)

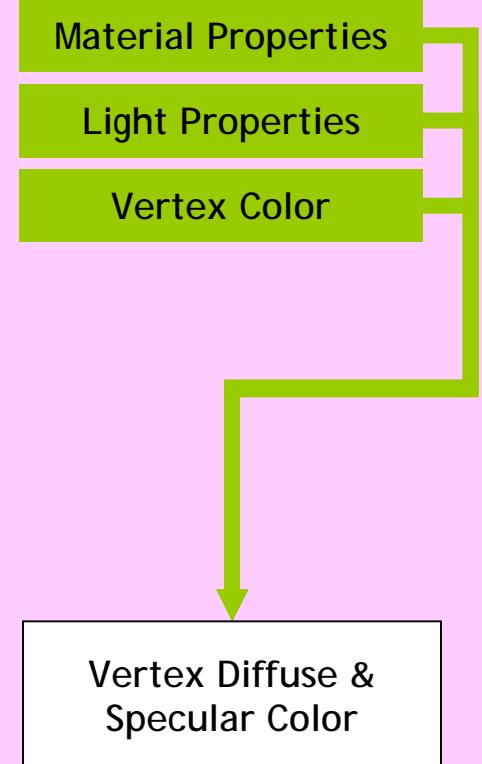


Transform and Lighting Unit

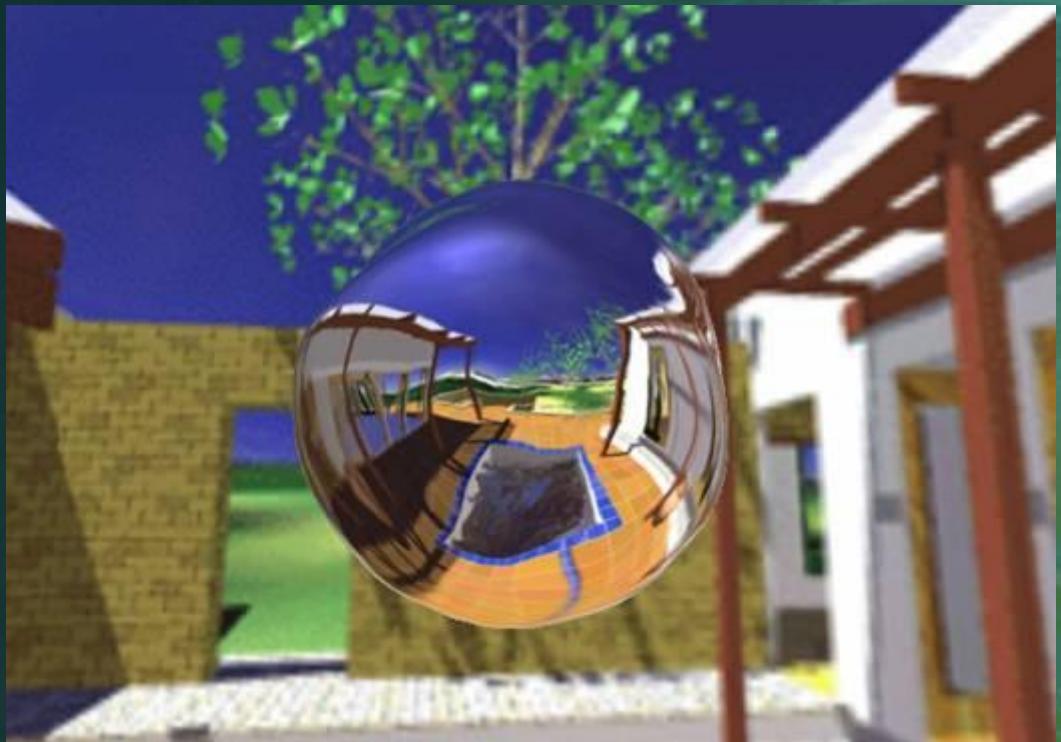
Transform



Lighting

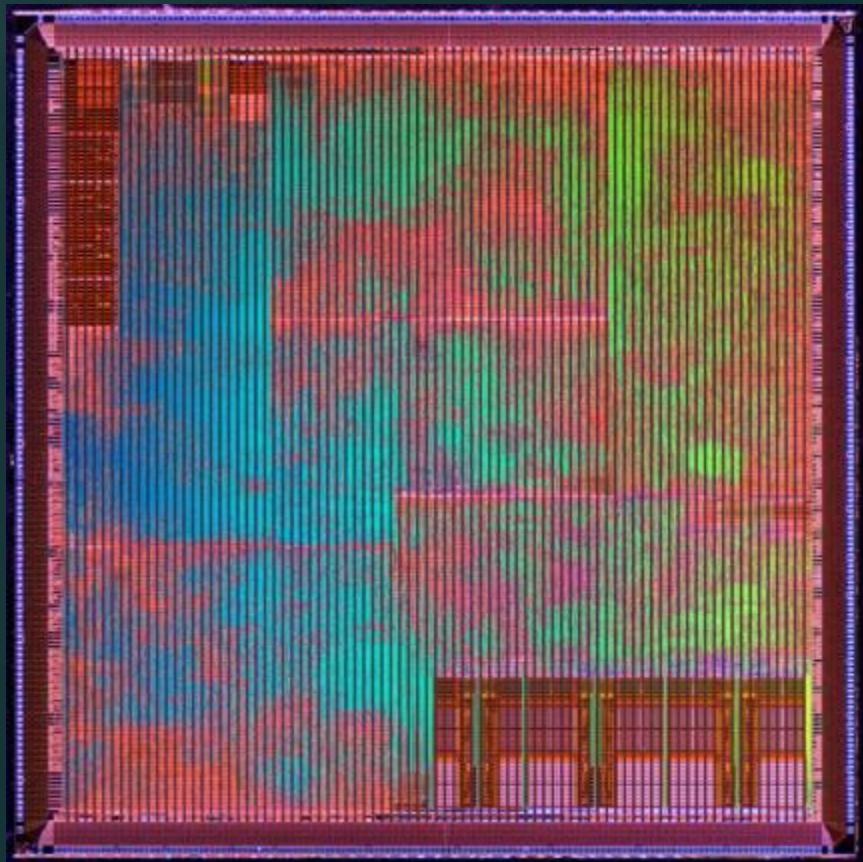


Wanda and Bubble



©2004 NVIDIA Corporation. All rights reserved.

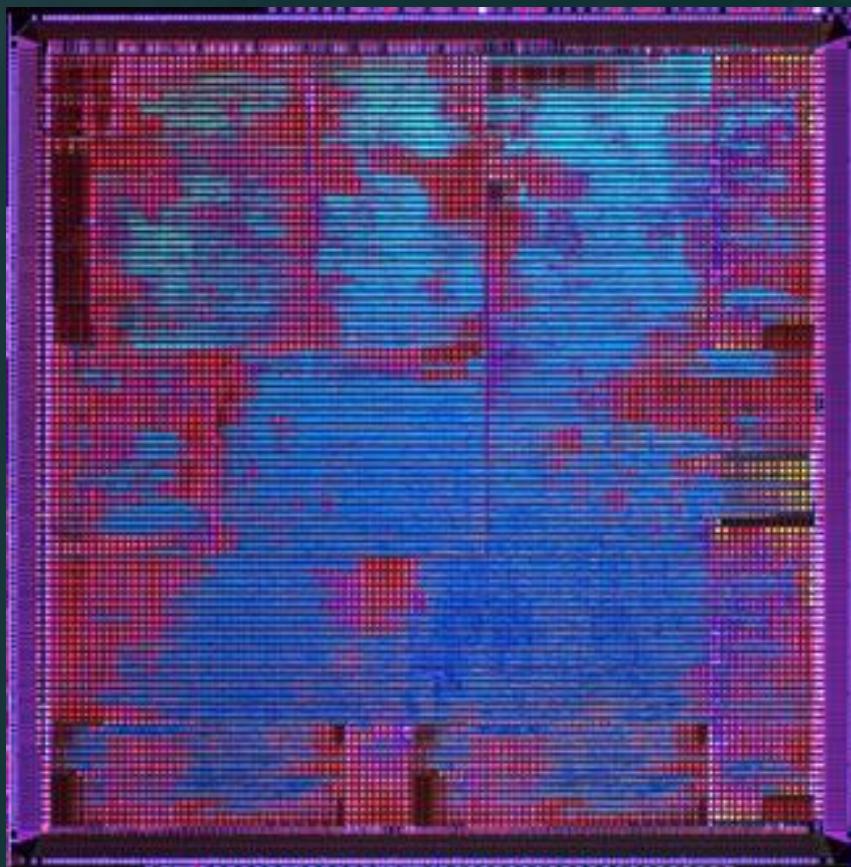
GeForce 2



**Pixel Shading
Multitexture**

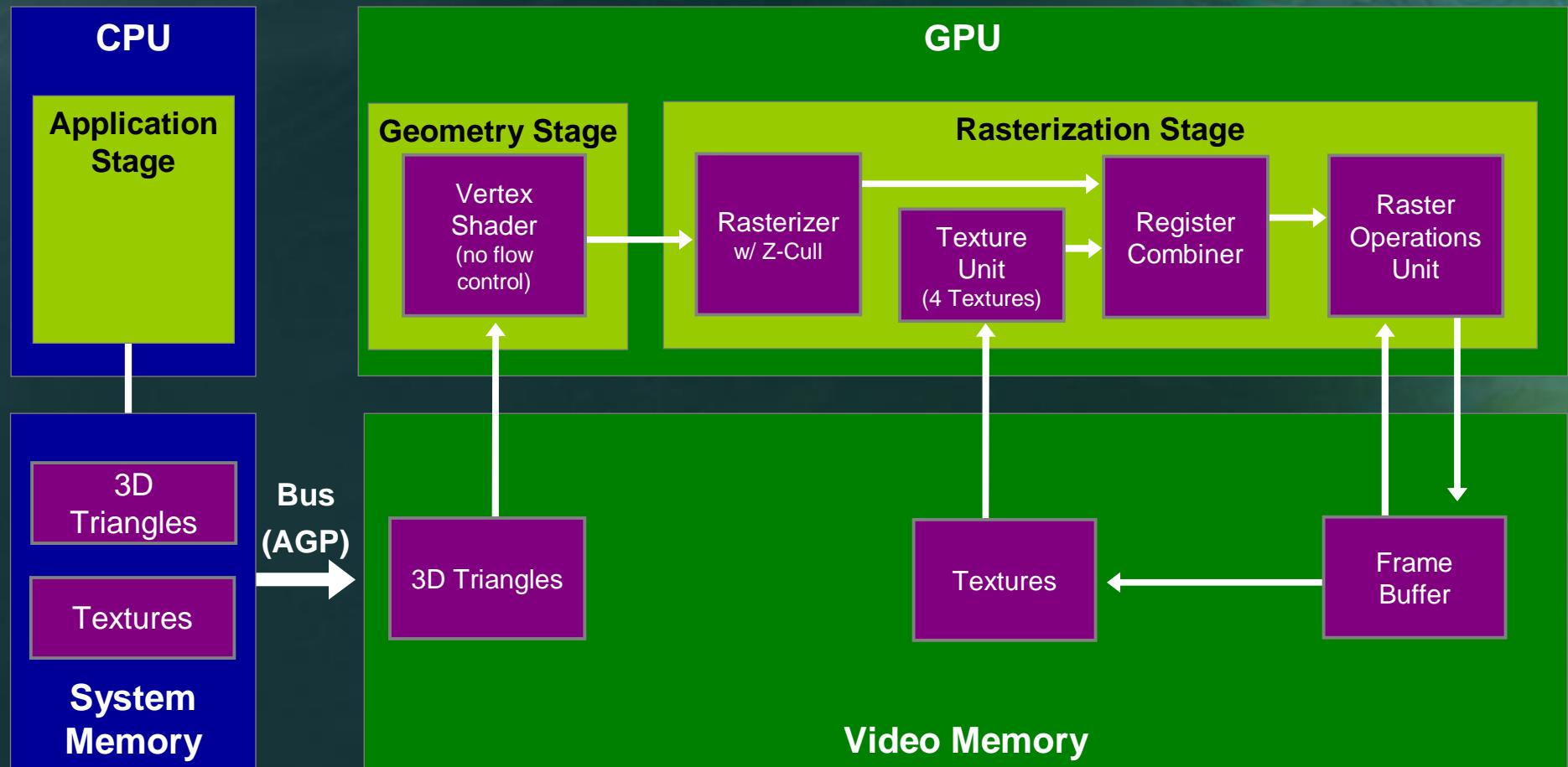


GeForce 3



**Programmable
Vertex Shading**

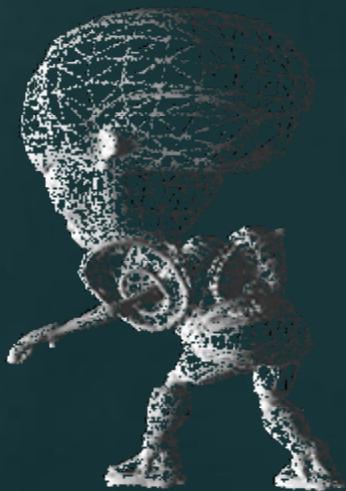
2001: Programmable Vertex Shader



Vertex Shader



- A programmable processor for any per-vertex computation



```
void VertexShader(
    // Input per vertex
    in float4 positionInModelSpace,
    in float2 textureCoordinates,
    in float3 normal,

    // Input per batch of triangles
uniform float4x4 modelToProjection,
uniform float3 lightDirection,

    // Output per vertex
out float4 positionInProjectionSpace,
out float2 textureCoordinatesOutput,
out float3 color
)
{
    // Vertex transformation
positionInProjectionSpace = mul(modelToProjection, positionInModelSpace);

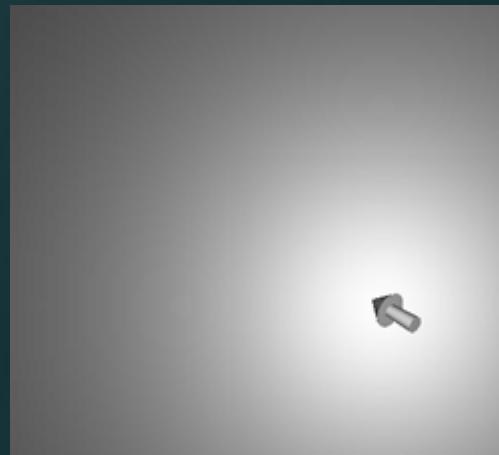
    // Texture coordinates copy
textureCoordinatesOutput = textureCoordinates;

    // Vertex color computation
color = dot(lightDirection, normal);
}
```

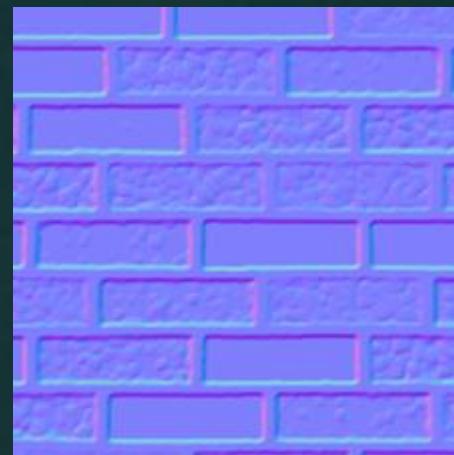
Bump Mapping



- Bump mapping involves fetching the per-pixel normal from a **normal map** texture (instead of using the interpolated vertex normal) in order to compute lighting at a given pixel



+



=



Diffuse light

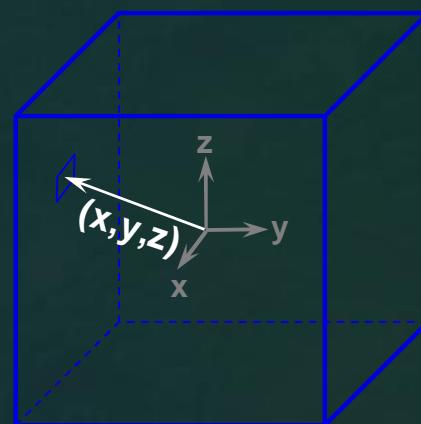
Normal Map

Diffuse light with bumps

Cubic Texture Mapping



Cubemap



Cubemap lookup
in direction (x, y, z)



Environment Mapping

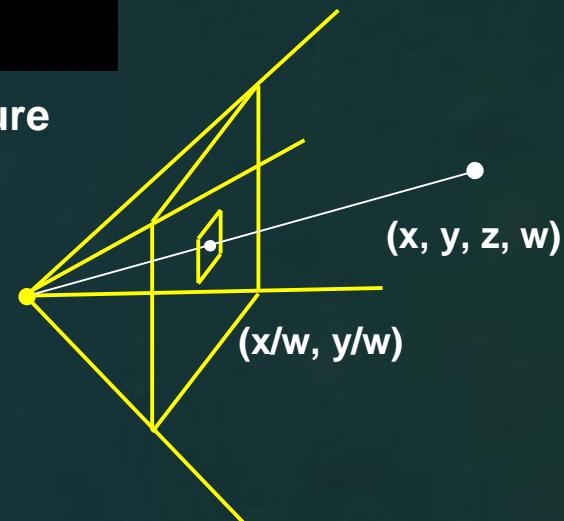
Reflection vector
is used to lookup
into the cubemap



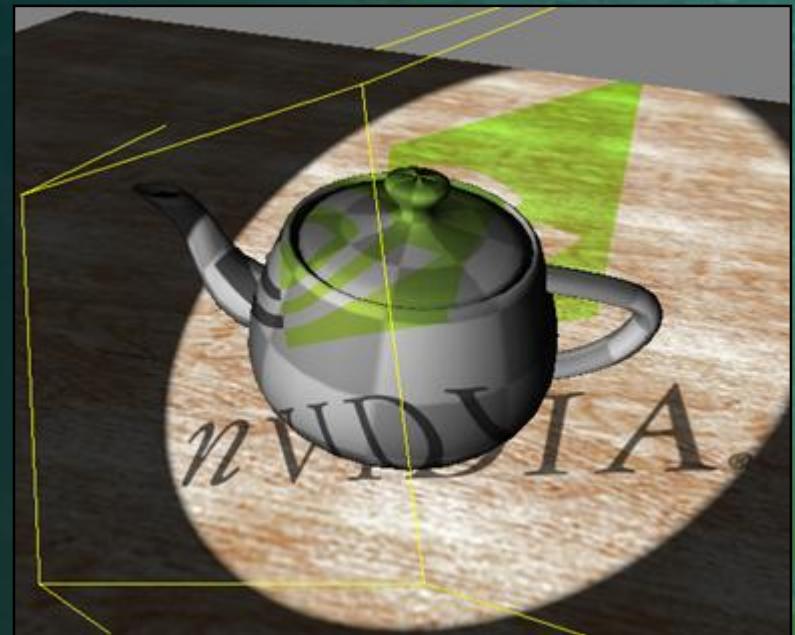
Projective Texture Mapping



Projected Texture

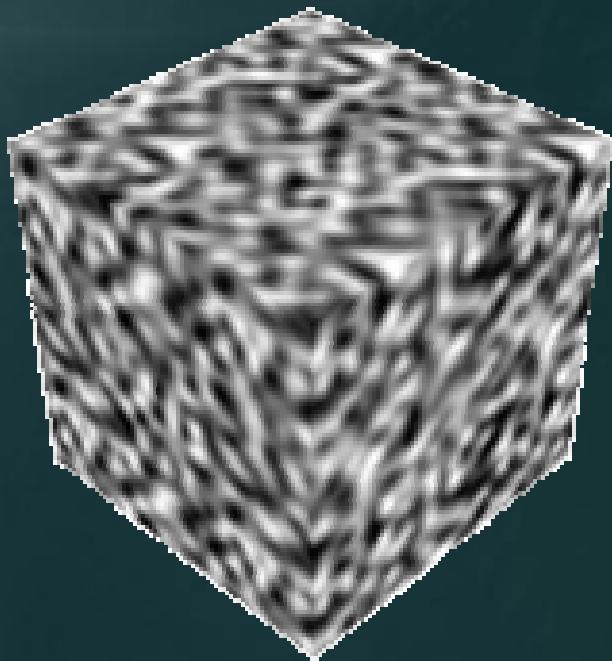


Projective Texture lookup

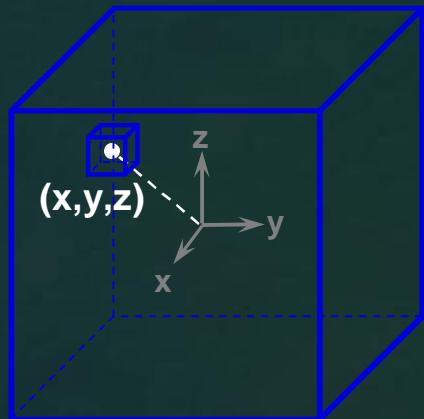


Texture Projection

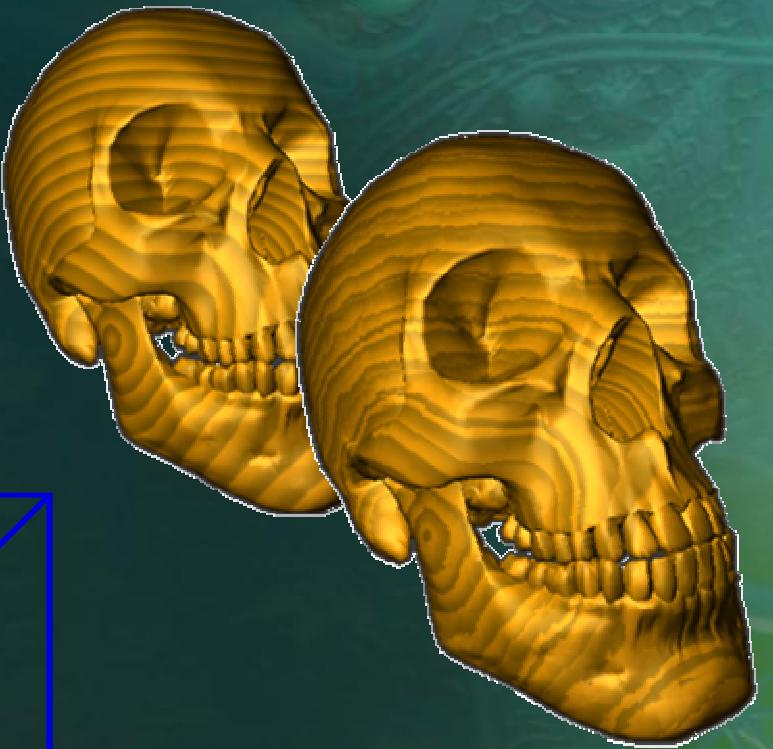
Volume Texture Mapping



Volume Texture
3D Noise



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



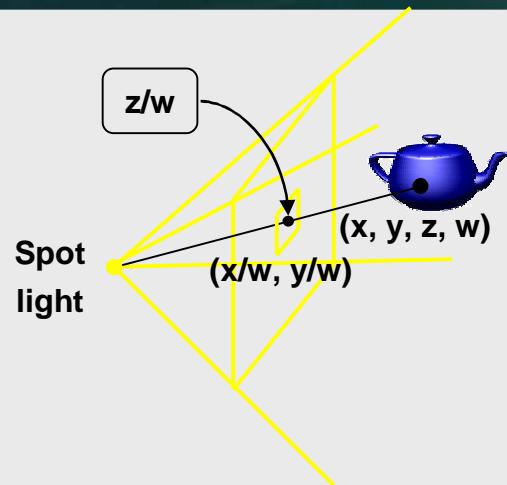
Solid Textures
Noise Perturbation

Hardware Shadow Mapping



Shadow Map Computation

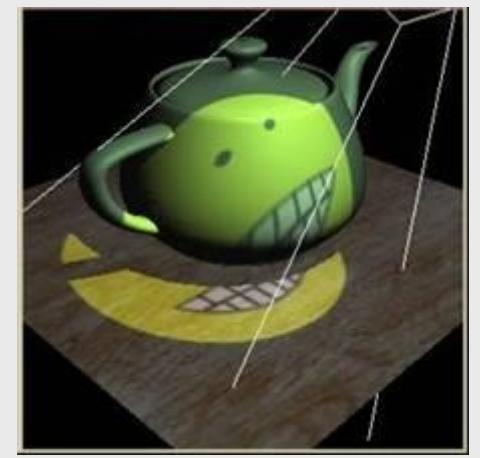
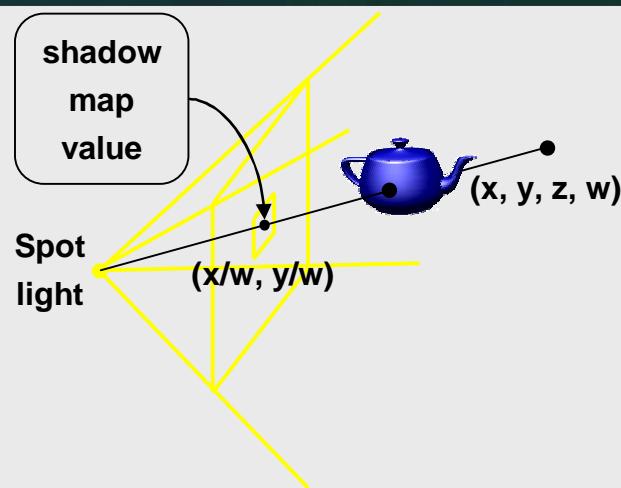
The shadow map contains the depth (z/w) of the 3D points visible from the light's point of view:



Shadow Rendering

A 3D point (x, y, z, w) is in shadow if:
 $z/w < \text{value of shadow map at } (x/w, y/w)$

A hardware shadow map lookup returns the value of this comparison between 0 and 1



Antialiasing: Examples



Text
Text

Text
Text

Antialiasing: Supersampling & Multisampling



- **Supersampling:**

Compute color and Z at higher resolution and display averaged color to smooth out the visual artifacts



- **Multisampling:**

Same thing except only Z is computed at higher resolution

- Multisampling performs antialiasing on primitive edges only



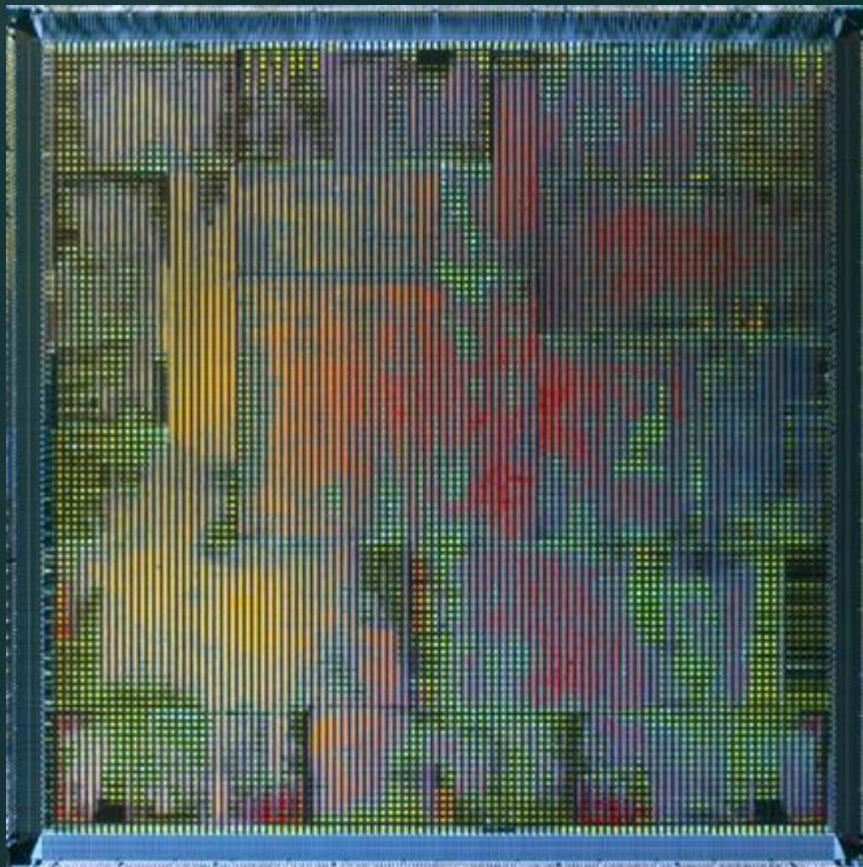
X-Isle: Dinosaur Isle



©2004 NVIDIA Corporation. All rights reserved.

Image courtesy of Crytek

GeForce 4



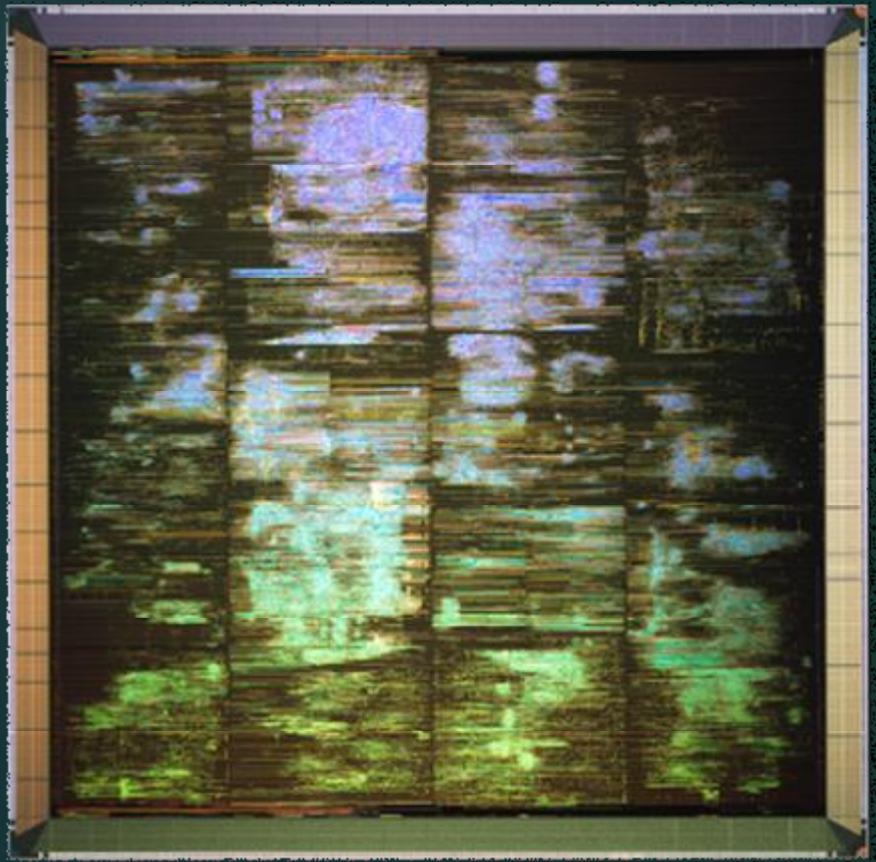
**Multiple
Vertex and
Pixel Shaders**

Wolfman



©2004 NVIDIA Corporation. All rights reserved.

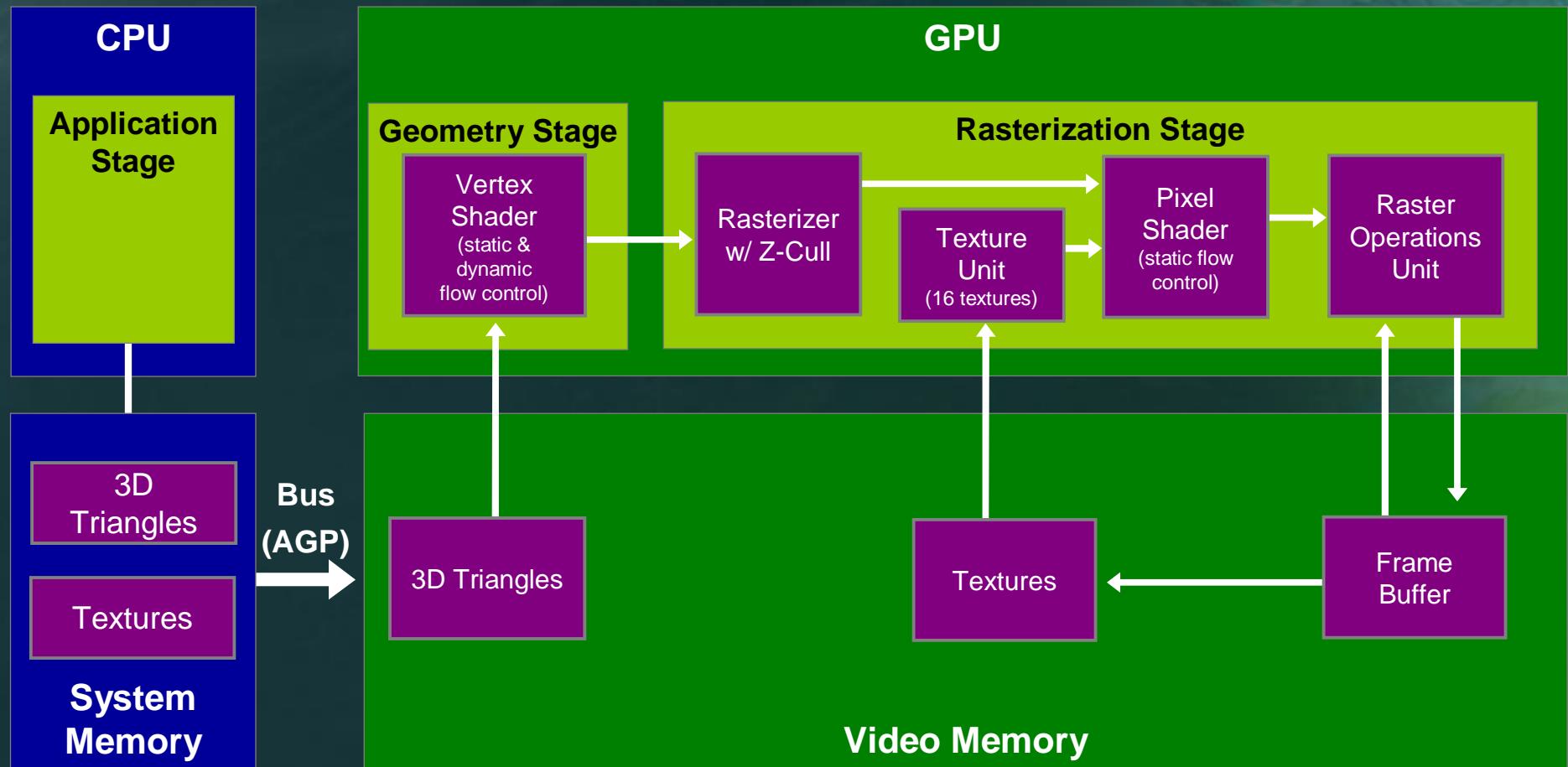
GeForce FX - >100M Transistors

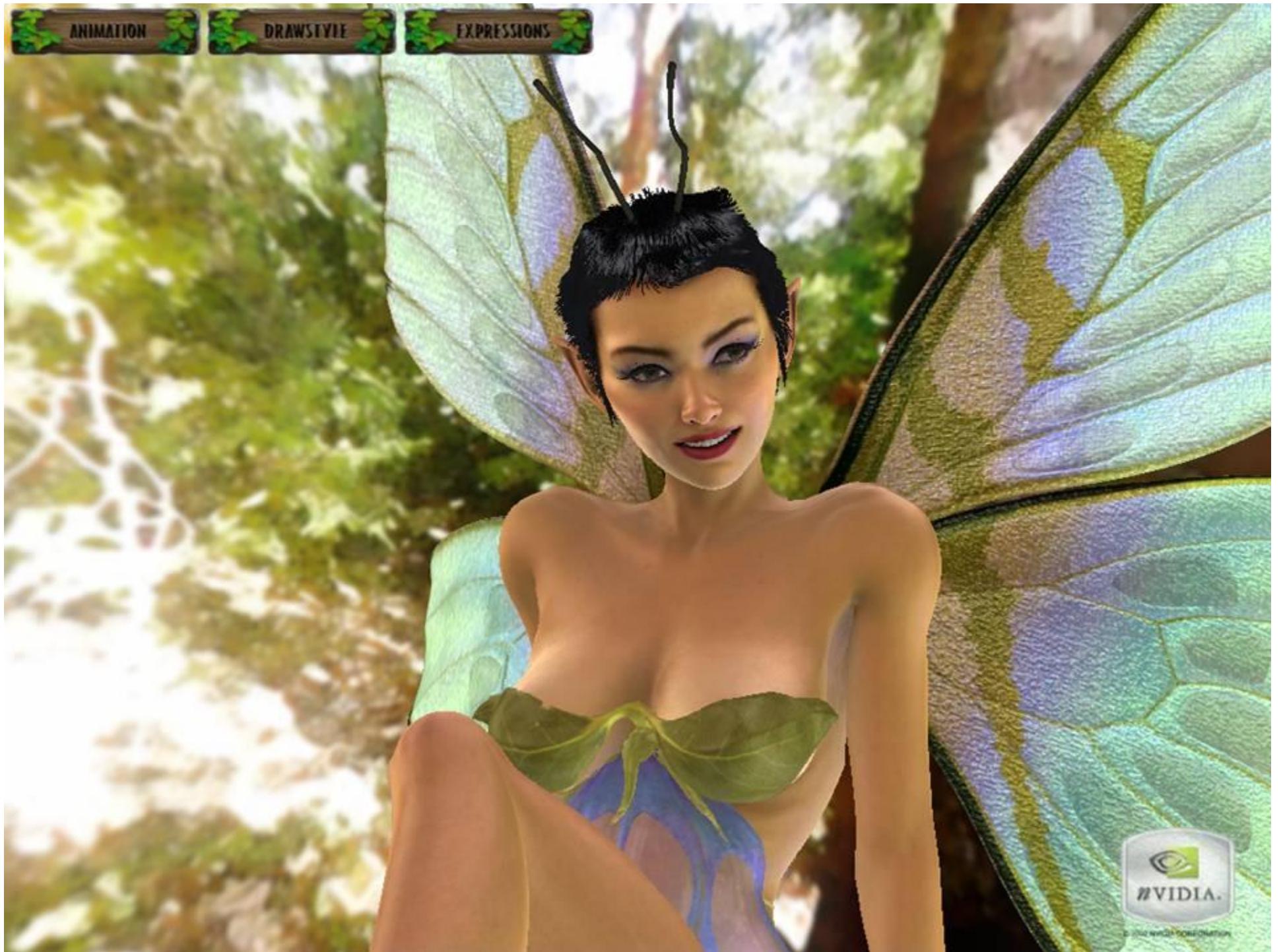


**Programmable
Pixel Shading
(DirectX 9.0)**

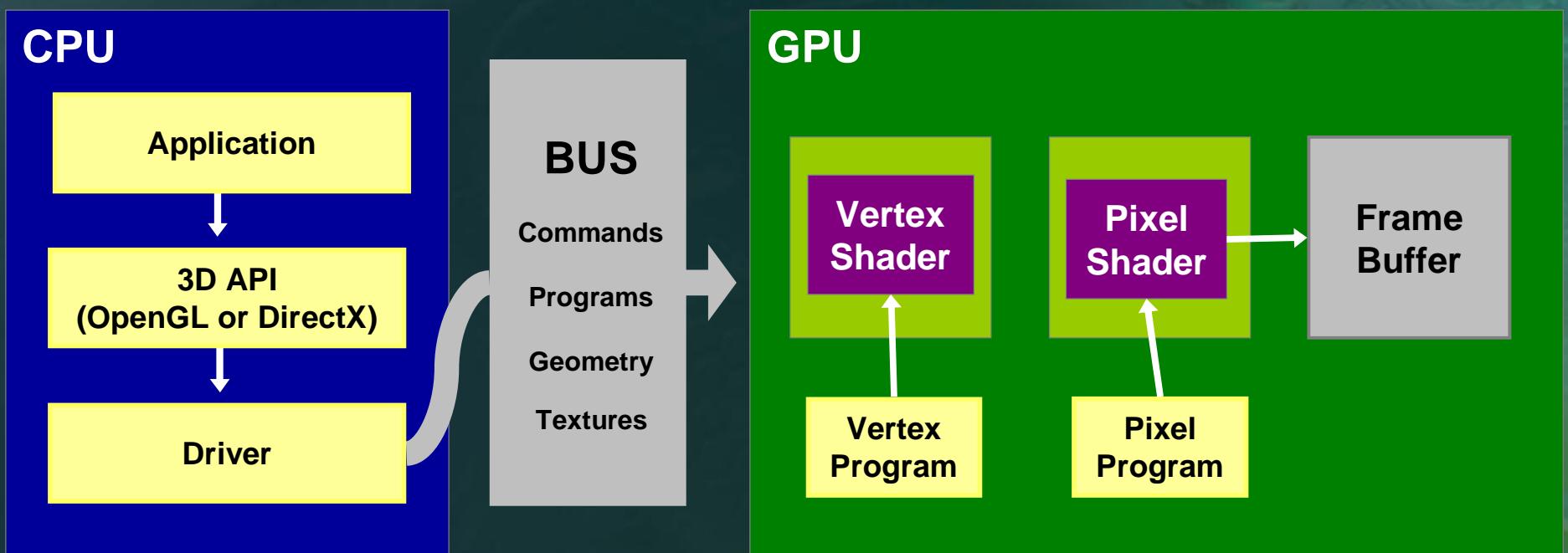
**Scalable
Architecture**

'02- '03: Programmable Pixel Shader





PC Graphics Software Architecture



- 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** (DirectX HLSL, OpenGL Shading Language, Cg)
- **Pushbuffer:** Contains the commands to be executed on the GPU



Pixel Shader

- A programmable processor for any per-pixel computation

```
void PixelShader(
    // Input per pixel
    in float2 textureCoordinates,
    in float3 normal,

    // Input per batch of triangles
    uniform sampler2D baseTexture,
    uniform float3 lightDirection,

    // Output per pixel
    out float3 color
)
{
    // Texture lookup
    float3 baseColor = tex2D(baseTexture,
        textureCoordinates);

    // Light computation
    float light = dot(lightDirection, normal);

    // Pixel color computation
    color = baseColor * light;
}
```

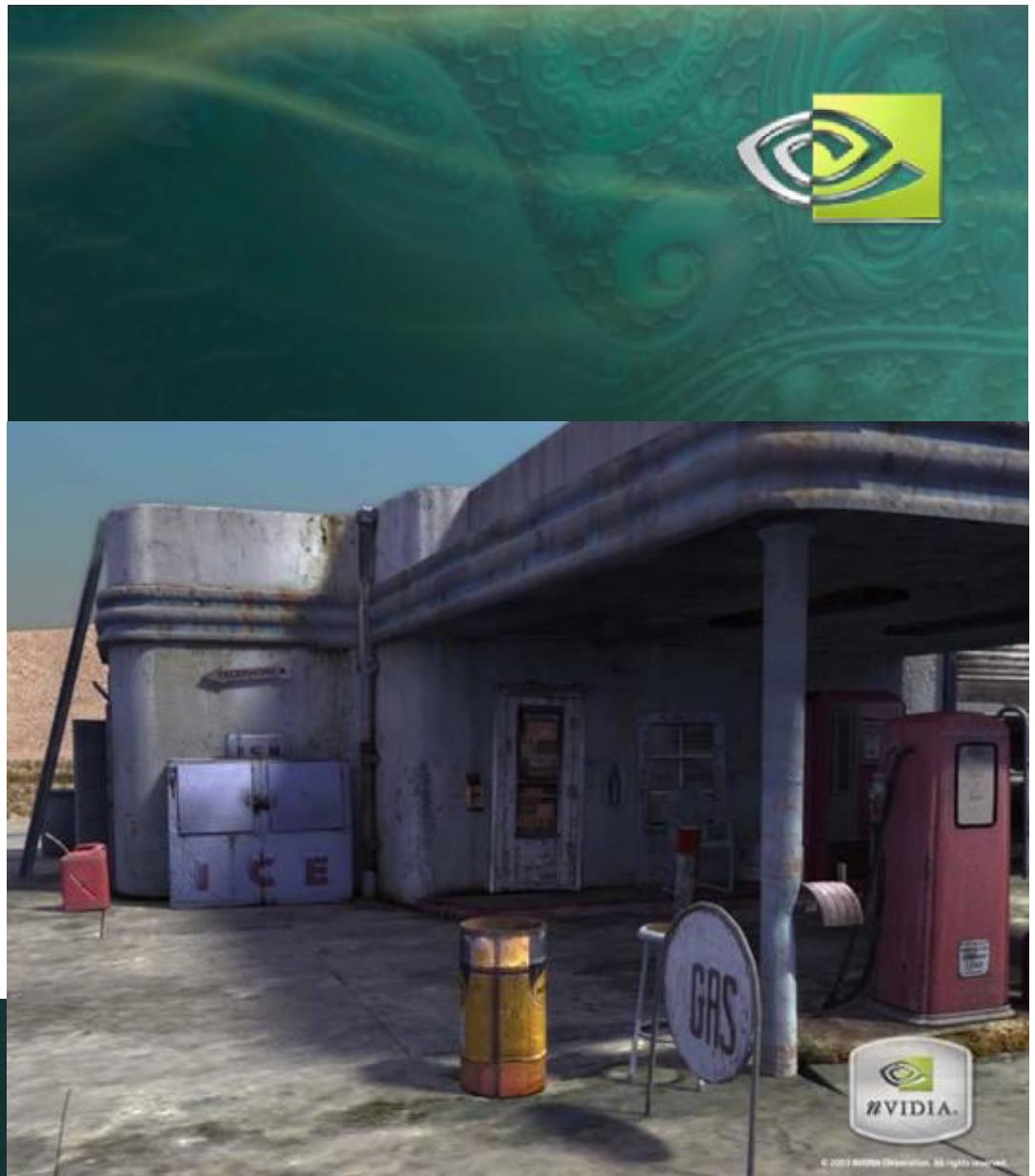


Shader: Static vs. Dynamic Flow Control

Static Flow Control
(condition varies
per batch of triangles)

Dynamic Flow Control
(condition varies
per vertex or pixel)

```
void Shader(
    ...
        // Input per vertex or per pixel
    in float3 normal,
    ...
        // Input per batch of triangles
    uniform float3 lightDirection,
    uniform bool computeLight,
    ...
)
{
    ...
    if (computeLight) {
        ...
        if (dot(lightDirection, normal)) {
            ...
        }
        ...
    }
    ...
}
```



©2004 NVIDIA Corporation. All rights reserved.

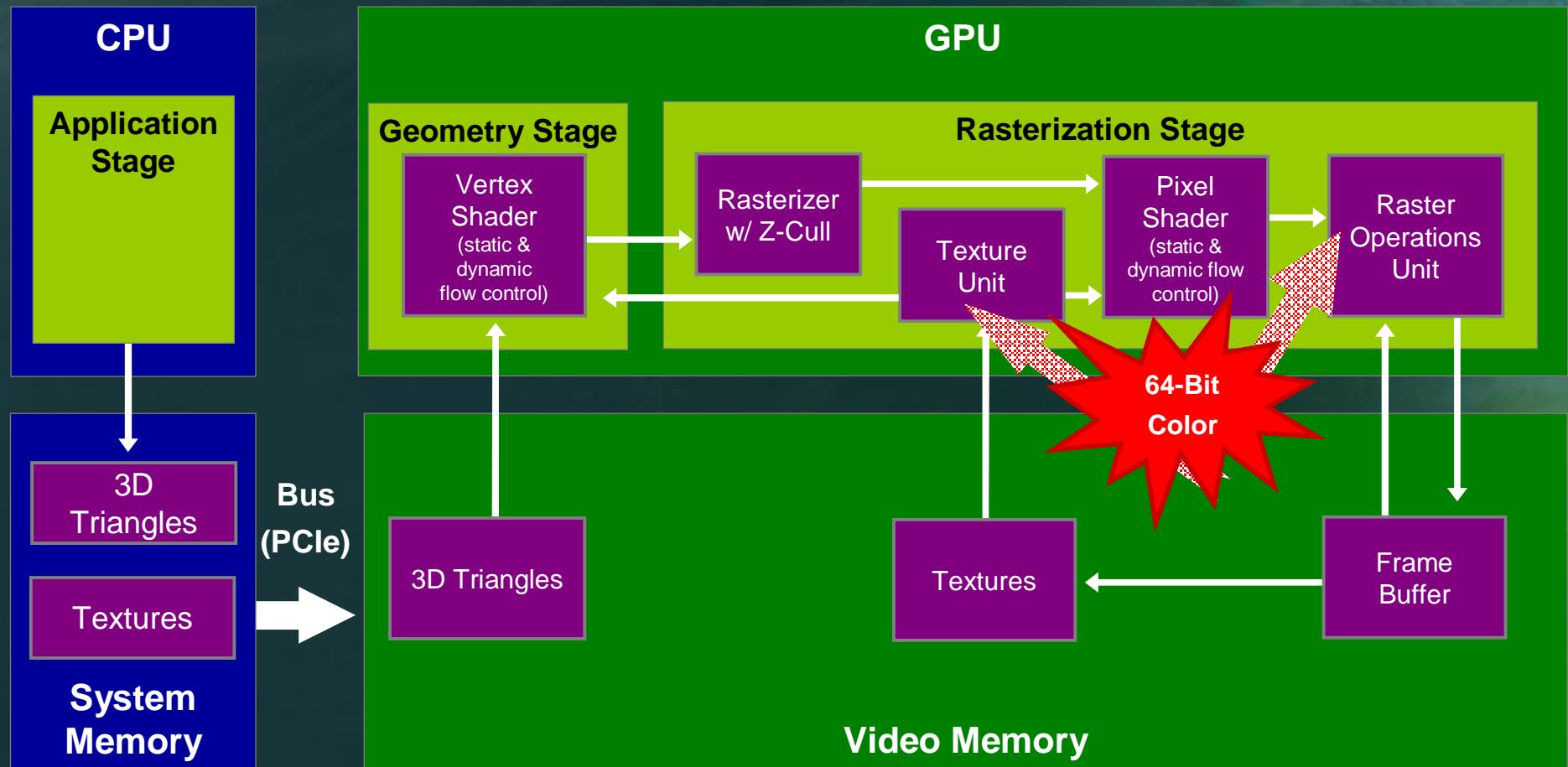
GeForce 6800 – 220M Transistors



**Shader Model
3.0**

**FP64 & High
Dynamic
Range**

2004: Shader Model 3.0 & 64-Bit Color Support



Shader Model 3.0



- Longer shaders → More complex shading
- Pixel shader:
 - Dynamic flow control → Better performance
 - Derivative instructions → Shader antialiasing
 - Support for 32-bit floating-point precision → Fewer artifacts
 - Face register → Faster two-sided lighting
- Vertex shader:
 - Texture access → Simulation on GPU, displacement mapping
- Geometry Instancing → Better performance

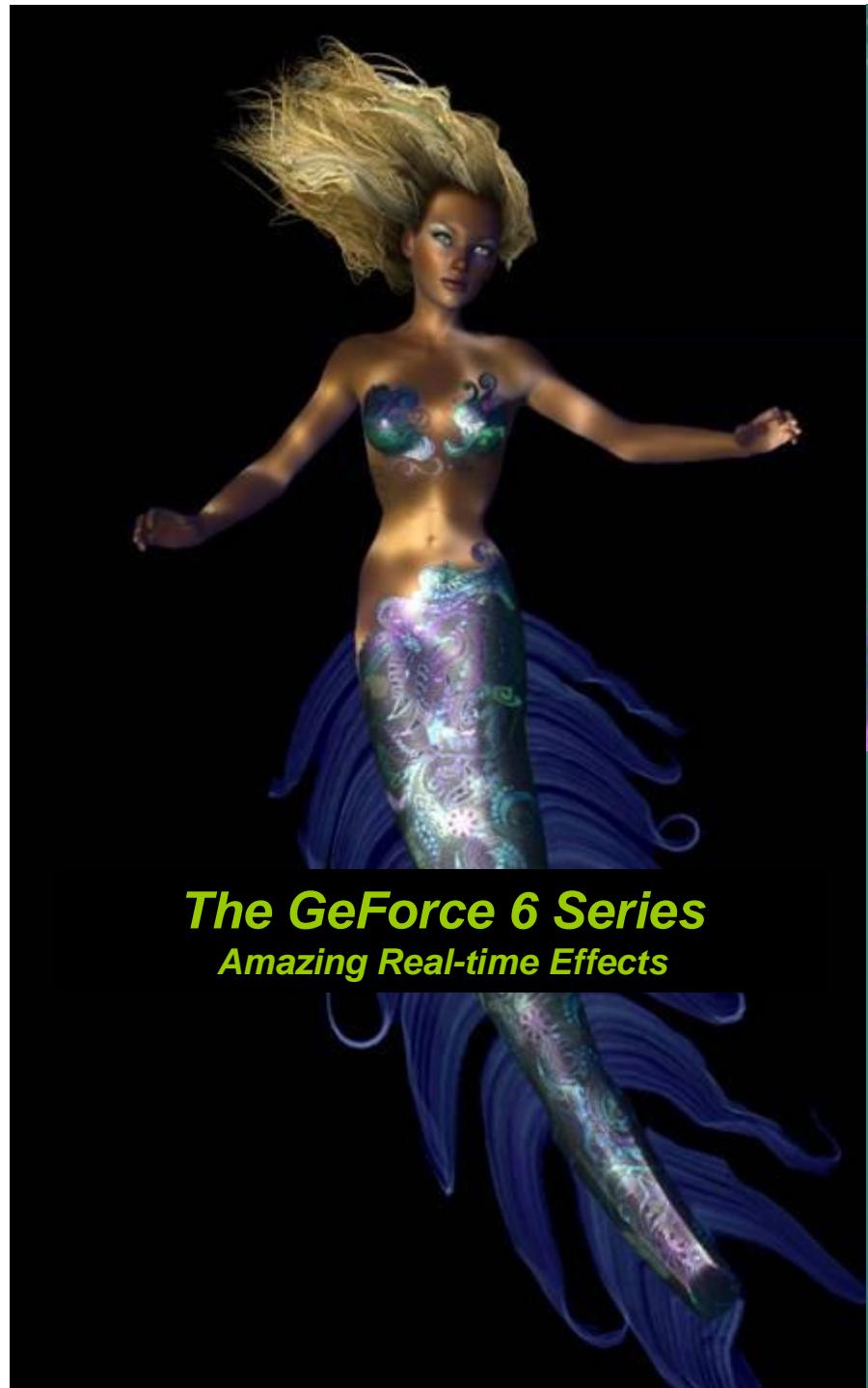


*Lord of the Rings™
The Battle for Middle-earth™*



Far Cry

©2004 NVIDIA Corporation. All rights reserved.



The GeForce 6 Series
Amazing Real-time Effects

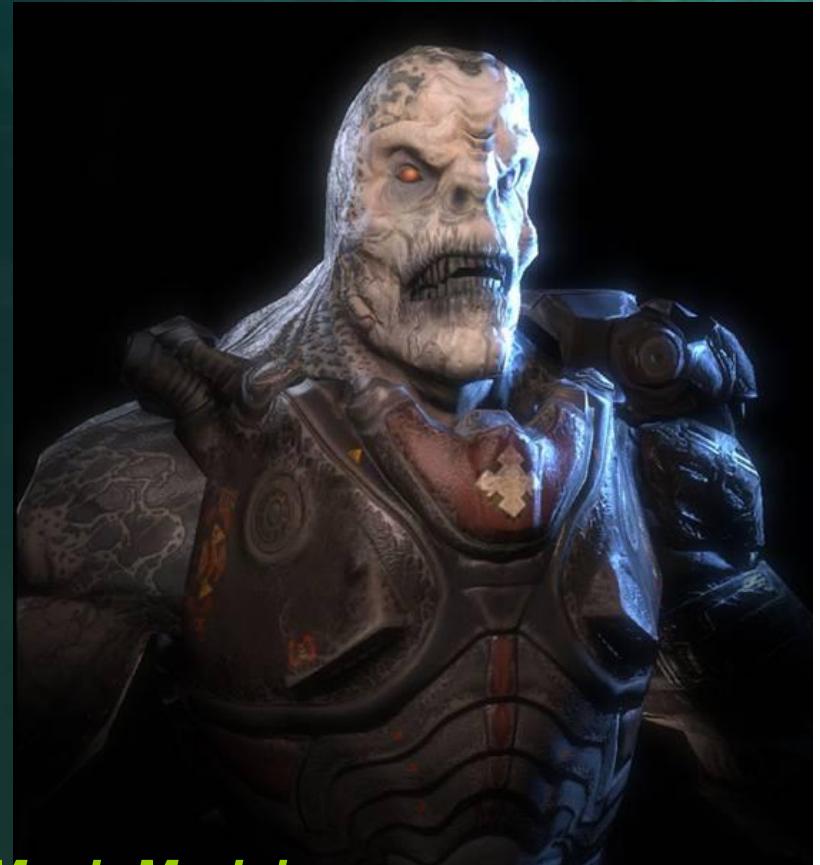




© 2004 NVIDIA Corporation. All rights reserved.

Shader Model 3.0 / 64-bit Floating Point Processing

Unreal Engine 3.0 Running On GeForce 6 Series



***2 Million Triangle Detail Mesh Models
High Dynamic Range Rendering
Fully Customizable Shaders***

Unreal Engine 3 Technology Demo © 2004 Epic Games. All Rights Reserved

UnReal Engine 3.0 Images Courtesy of Epic Games

Unreal Engine 3 Technology Demo © 2004 Epic Games. All Rights Reserved

Shader Model 3.0 / 64-bit Floating Point Processing

Unreal Engine 3.0 Running On GeForce 6 Series

*100 Million Triangle Source Content Scene
High Dynamic Range Rendering*

UnReal Engine 3.0 Images Courtesy of Epic Games



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
- HDR Rendering Engine -
 - Compute surface reflectance, save in HDR buffer
 - Contributions from multiple lights are additive (blended)
 - Add image-space special effects & Post to HDR buffer
 - AA, Glow, Depth of Field, Motion Blur

©2004 NVIDIA Corporation. All rights reserved.



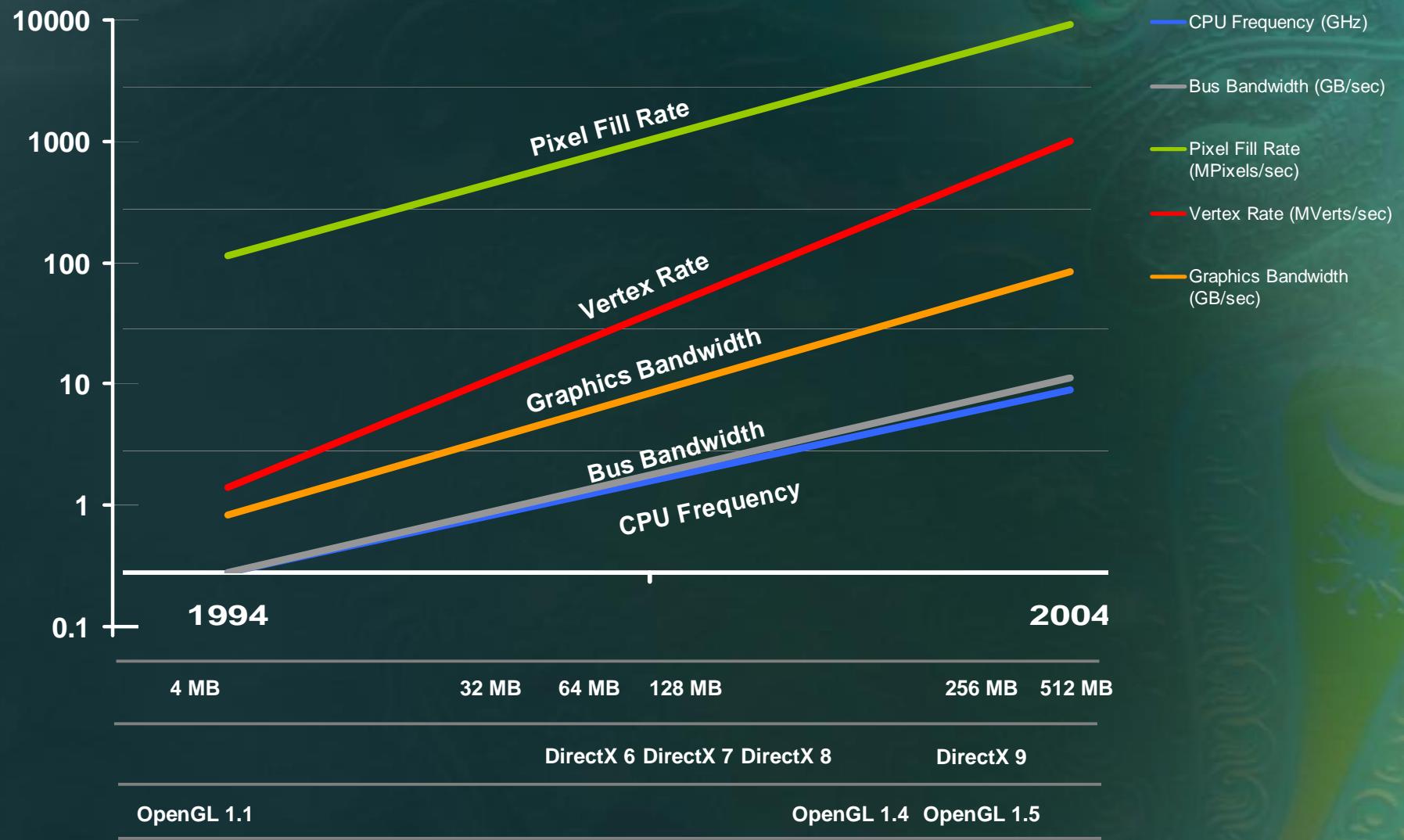
Real-Time Tone Mapping

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



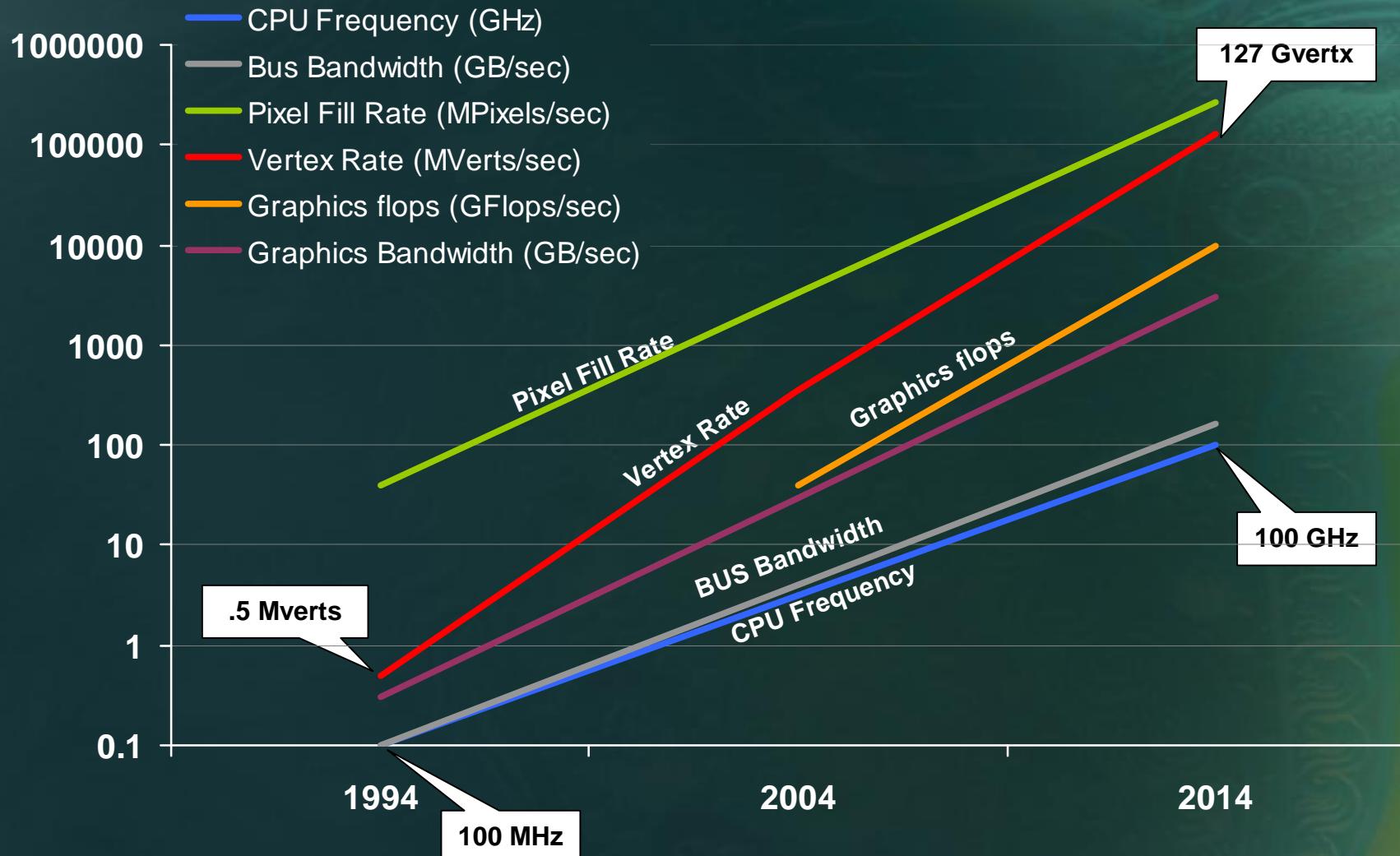
Renderings of the same scene, from low to high exposure

Evolution of Performance



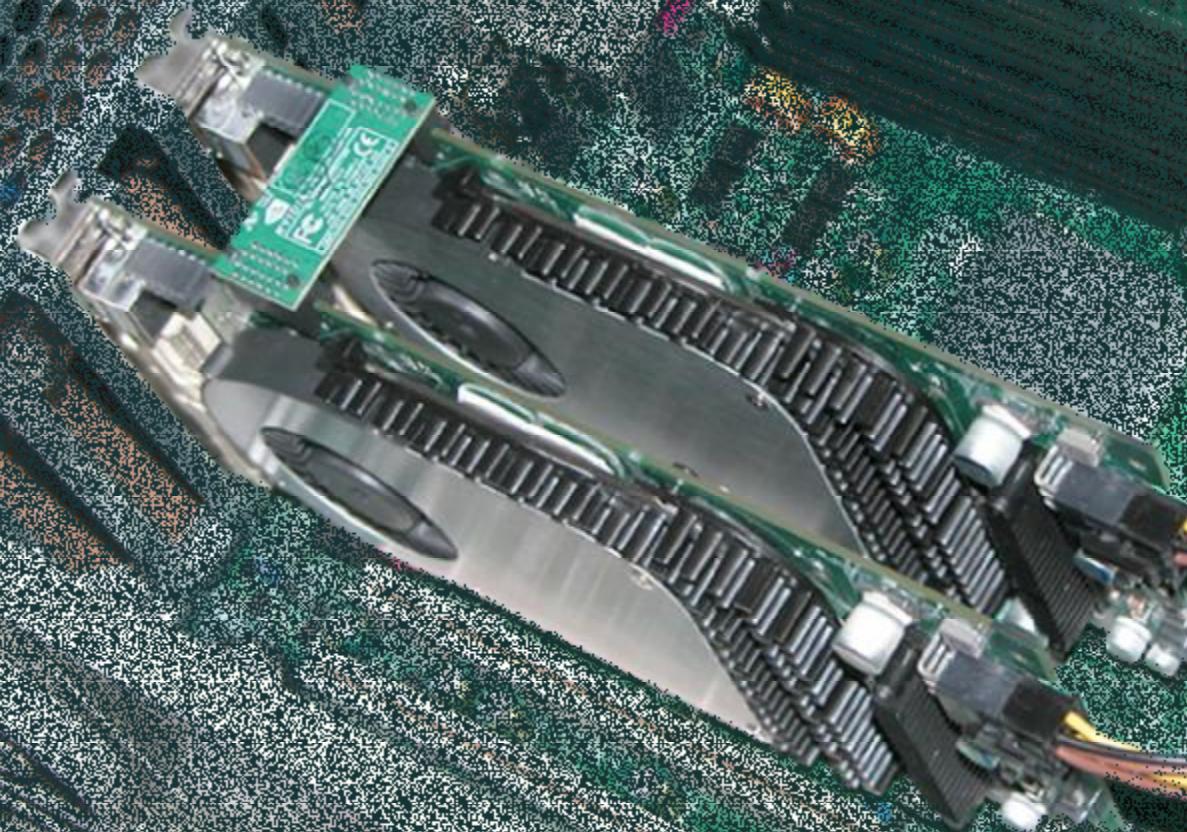
©2004 NVIDIA Corporation. All rights reserved.

Looking Ahead: Now + 10 years



©2004 NVIDIA Corporation. All rights reserved.

NVIDIA SLI Multi-GPU



©2004 NVIDIA Corporation. All rights reserved.

Progression of Graphics



Virtual Fighter
NV1
1Mtrans



Wanda
NV1x
22Mtrans



Wolfman
NV2x
63Mtrans



Dawn
NV3x
130Mtrans



Nalu
NV4x
222M

©2004 NVIDIA Corporation. All rights reserved.



A woman with long, dark hair and bright green eyes is shown from the chest up. She has a large, intricate tattoo on her right shoulder and arm, featuring swirling patterns in shades of blue, purple, and gold. The background is a dark, textured surface.

Thank You