Introduction to Cg

Cg

- C for Graphics
- Programming language specialized for graphics
- Defined, Implemented and supported by NVIDIA
  - Examples of other languages:
    - Microsoft High-Level Shader Language
    - OpenGL Shading Language
- Exploits the specialized, high performance nature of GPUs
- Offers control over shape, appearance and motion of objects rendered using GPUs
- More than just shading (physical simulation, compositing, etc.)

Cg's Data-Flow Model

- Cg is based on a data-flow computation model
- Computation occurs in response to data that flows through a sequence of processing steps
- Every time a vertex is processed, a vertex Cg program executes
- Every time the rasterizer generates a fragment, a fragment Cg program executes
- Cg programs execute on the GPU

Cg and Other Languages

- Cg coexist with other languages such as C or C++
- Applications supply Cg programs to instruct the GPU on how to accomplish programmable rendering effects that cannot be performed on the CPU at the same rates

Examples of Cg Shaders

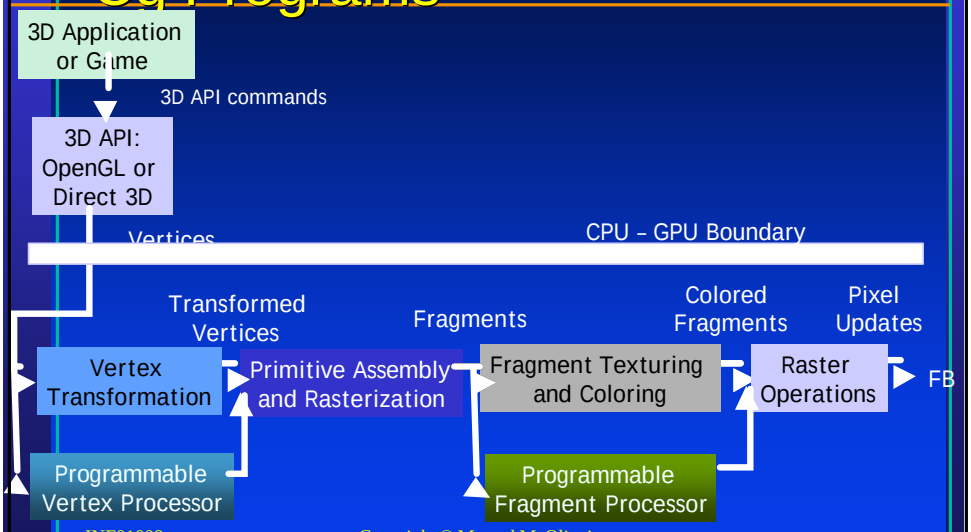
Cg and Other Languages

- Cg is much simpler than C or C++
  - Does not support classes
  - Currently does not support pointers or memory allocation
  - Does not support file input/output operations
- Cg provides arrays, structures, loops, conditionals and function calls
- Cg has a library of functions (standard library)
  - Functions like normalize, reflect, lit, dot, etc.

Cg Programs

- Cg programs execute in relative isolation
- The processing of a vertex/fragment has not effect on other vertices/fragments processed at the same time
- Well suited for hardware execution by highly pipelined and parallel hardware

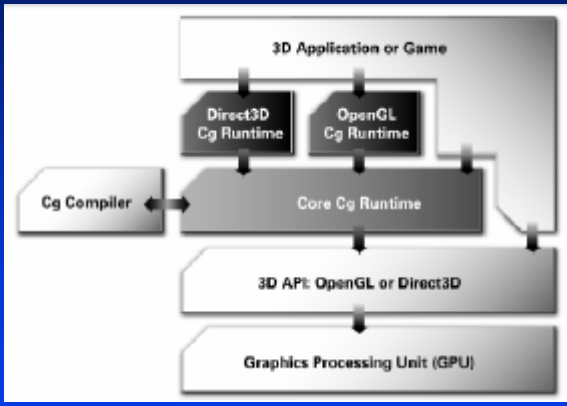
Cg Programs



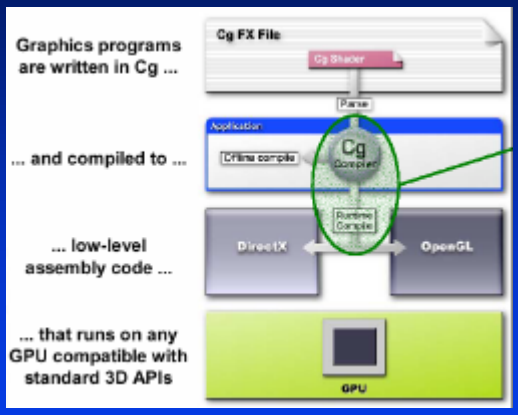
The Cg Environment

- Cg Compiler and Runtime
  - Translates a Cg program into a form accepted by the 3D programming interface (either OpenGL or Direct3D)
  - Core set of functions and structures encapsulating the 3D API-independent functionality of the runtime
- API-specific Cg Libraries (CgGL and CgD3D)
  - Transfer the OpenGL or Direct3D translation of the original Cg program to the GPU using the appropriate OpenGL or Direct3D commands
  - OpenGL/Direct3D-specific set of functions built on top of the core set
- OpenGL/Direct3D Driver
  - Performs the final translation into the GPU machine language

The Cg Environment



The Cg Environment



Vertex Program Example

```
struct v_output {
    float4 position : POSITION;
    float4 color    : COLOR;
}

v_output v_green(float2 position : POSITION)
{
    v_output OUT;

    OUT.position = float4(position, 0, 1);
    OUT.color  = float4(0, 1, 0, 1);   // RGBA green

    return OUT;
}
```


Fragment Program Example

```
struct f_output {
    float4 color      : COLOR;
}

f_output  f_passthrough(float4 color : COLOR)
{
    f_output OUT;

    OUT.color  = color;

    return OUT;
}
```


Compiling and Loading Cg