CMSC 491G/691G

Computer Graphics for Games

Marc Olano

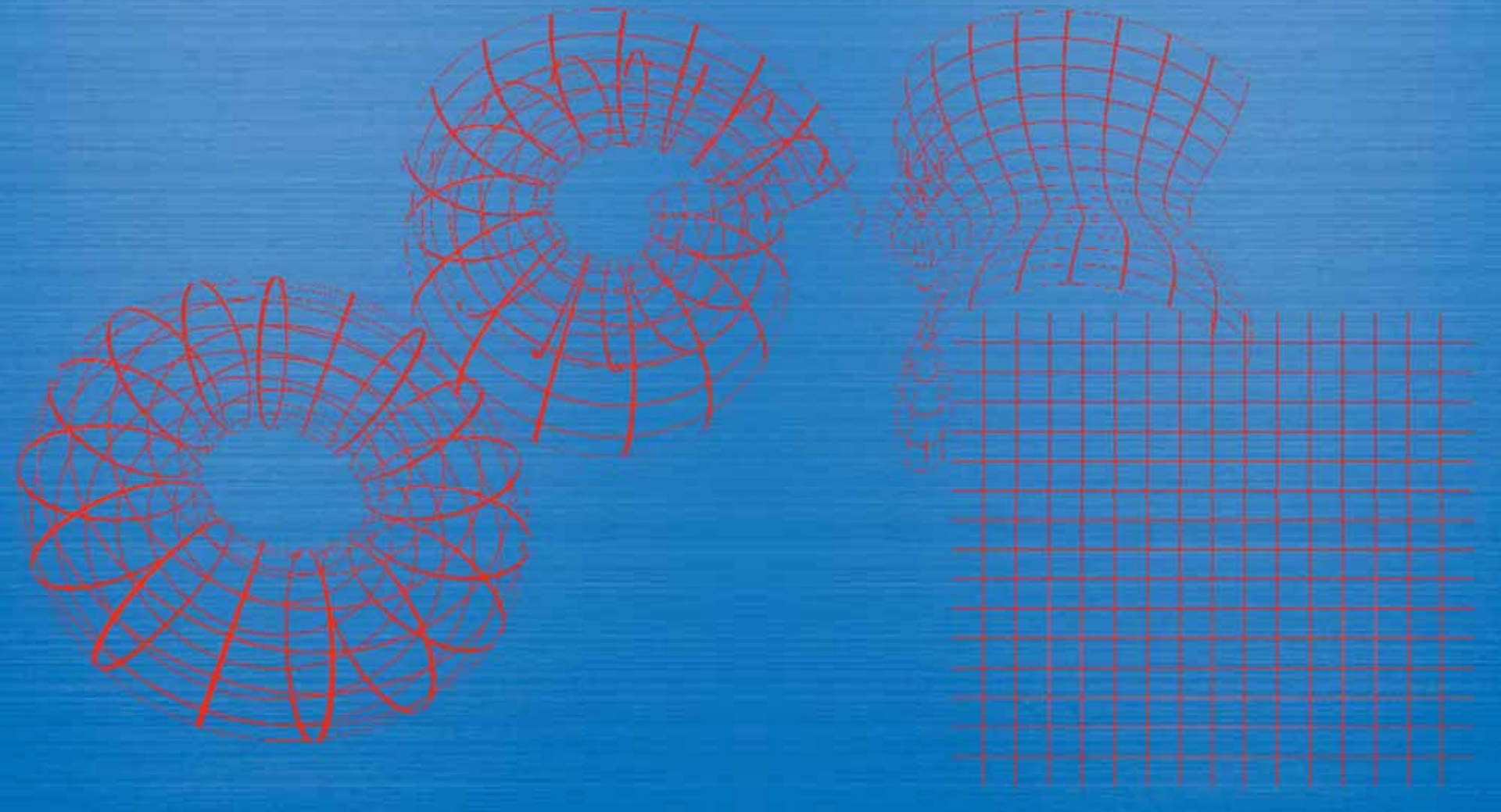
OpenGL Shading

- High level language
  - OpenGL Shading Language = GLslang = GLSL
- Integrated into OpenGL API
  - no extra run-time

API-integrated

- Compiler built into driver
  - Presumably they know your card best
  - IHV's must produce (good) compilers
- Use built-in parameters (glColor, glNormal, …)
  - Add your own
  - Built-ins disappearing in OpenGL *"Mount Evans"*
- Other options can still produce low-level code
  - Cg, ASHLI, RapidMind, …
  - With loss of integration

Vertex Demo: Blend Positions



High-level Code

```
void main() {
    float Kin = gl_Color.r;          // key input

    // screen position from vertex and texture
    vec4 Vp = ftransform();
    vec4 Tp = vec4(gl_MultiTexCoord0.xy*1.8-.9, 0.,1.);

    // interpolate between Vp and Tp
    gl_Position = mix(Tp,Vp,pow(1.-Kin,8.));

    // copy to output
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_TexCoord[1] = Vp;
    gl_TexCoord[3] = vec4(Kin);
}
```

Main Function

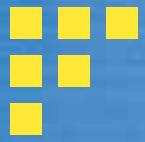
```
void main() {
    float Kin = gl_Color.r;          // key input

    // screen position from vertex and texture
    vec4 Vp = ftransform();
    vec4 Tp = vec4(gl_MultiTexCoord0.xy*1.8-.9, 0.,1.);

    // interpolate between Vp and Tp
    gl_Position = mix(Tp,Vp,pow(1.-Kin,8.));

    // copy to output
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_TexCoord[1] = Vp;
    gl_TexCoord[3] = vec4(Kin);
}
```

Use Standard OpenGL State

```
void main() {
    float Kin = gl_Color.r;          // key input

    // screen position from vertex and texture
    vec4 Vp = ftransform();
    vec4 Tp = vec4(gl_MultiTexCoord0.xy*1.8-.9, 0.,1.);

    // interpolate between Vp and Tp
    gl_Position = mix(Tp,Vp,pow(1.-Kin,8.));

    // copy to output
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_TexCoord[1] = Vp;
    gl_TexCoord[3] = vec4(Kin);
}
```

Built-in Types

```
void main() {
    float Kin = gl_Color.r;          // key input

    // screen position from vertex and texture
    vec4 Vp = ftransform();
    vec4 Tp = vec4(gl_MultiTexCoord0.xy*1.8-.9, 0.,1.);

    // interpolate between Vp and Tp
    gl_Position = mix(Tp,Vp,pow(1.-Kin,8.));

    // copy to output
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_TexCoord[1] = Vp;
    gl_TexCoord[3] = vec4(Kin);
}
```



Swizzle / Channel Selection

```
void main() {
    float Kin = gl_Color.r;         // key input

    // screen position from vertex and texture
    vec4 Vp = ftransform();
    vec4 Tp = vec4(gl_MultiTexCoord0.xy*1.8-.9, 0.,1.);

    // interpolate between Vp and Tp
    gl_Position = mix(Tp,Vp,pow(1.-Kin,8.));

    // copy to output
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_TexCoord[1] = Vp;
    gl_TexCoord[3] = vec4(Kin);
}
```

Vector Construction

```
void main() {
    float Kin = gl_Color.r;          // key input

    // screen position from vertex and texture
    vec4 Vp = ftransform();
    vec4 Tp = vec4(gl_MultiTexCoord0.xy*1.8-.9, 0.,1.);

    // interpolate between Vp and Tp
    gl_Position = mix(Tp,Vp,pow(1.-Kin,8.));

    // copy to output
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_TexCoord[1] = Vp;
    gl_TexCoord[3] = vec4(Kin);
}
```

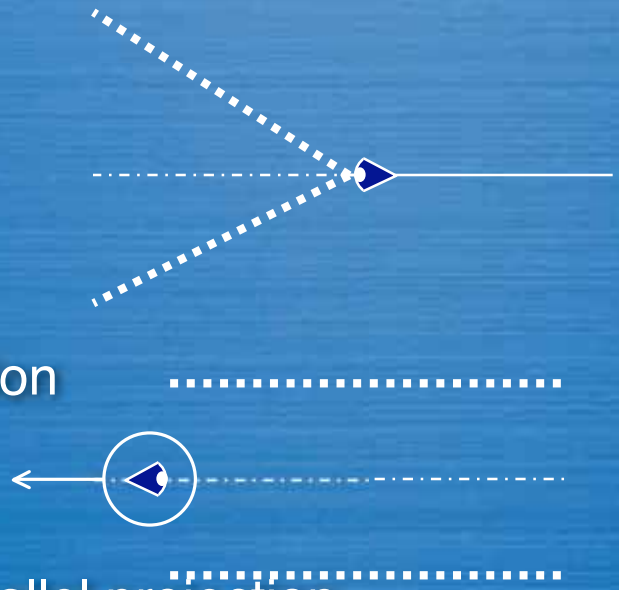
Built-in Functions

```
void main() {
    float Kin = gl_Color.r;          // key input

    // screen position from vertex and texture
    vec4 Vp = ftransform();
    vec4 Tp = vec4(gl_MultiTexCoord0.xy*1.8-.9, 0.,1.);

    // interpolate between Vp and Tp
    gl_Position = mix(Tp,Vp,pow(1.-Kin,8.));

    // copy to output
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_TexCoord[1] = Vp;
    gl_TexCoord[3] = vec4(Kin);
}
```

Vertex + Fragment Demo: Fresnel Environment Map

Trick #1: Where is the Eye

- Where is the Eye in Eye Space?
  - (0,0,0)? Not necessarily!
- **Know** where it is in Clip Space
  - (0,0,-1,0), looking in the (0,0,1,0) direction
    - Assuming GL_LESS depth test
  - Invert projection to find the eye!
  - Works for any eye position, or even parallel projection.



Trick #2: Subtract Homogeneous Points

- Homogeneous point: vec4(V.xyz, V.w)
  - 3D equivalent: V.xyz/V.w
  - Defers division, makes perspective, translation, and many things happy
- Vector subtraction: V–E
  - V.xyz/V.w – E.xyz/E.w
  - (V.xyz*E.w – E.xyz*V.w)/(V.w*E.w)

Trick #3: Skip Division for Normalize

- normalize(V.xyz/V.w) = normalize(V.xyz)
  - If V.w isn't negative
- Put it all together:
  - normalize(V-E)
  - = normalize(V.xyz*E.w - E.xyz*V.w)