

ECE4893A/CS4803MPG:

# MULTICORE AND GPU PROGRAMMING FOR VIDEO GAMES



## Animation on the GPU

Prof. Aaron Lanterman



School of Electrical and Computer Engineering

Georgia Institute of Technology



# Typical keyframed animations

- Standing
- Running
- Kneeling
- Ducking
- Attacking
- Passing on, ceasing to be, expiring and going to meet maker, pushing up the daisies, kicking the bucket, shuffling off mortal coil, running down the curtain, joining the invisible choir, becoming an ex-parrot

# Basic keyframing (vertex shader)

```
void C6E3v_keyFrame(float3 positionA : POSITION,  
                   float3 positionB : TEXCOORD1,  
                   float4 color      : COLOR,  
                   float2 texCoord   : TEXCOORD0,  
  
                   out float4 oPosition : POSITION,  
                   out float2 oTexCoord : TEXCOORD0,  
                   out float4 oColor    : COLOR,  
  
                   uniform float      keyFrameBlend,  
                   uniform float4x4   modelViewProj)  
{  
    float3 position = lerp(positionA, positionB,  
                           keyFrameBlend);  
    oPosition = mul(modelViewProj, float4(position, 1));  
    oTexCoord = texCoord;  
    oColor = color;  
}
```

# A Light structure

```
struct Light {  
    float3 eyePosition;    // In object space  
    float3 lightPosition; // In object space  
    float4 lightColor;  
    float  specularExponent;  
    float  ambient;  
};
```

# Keyframing with lighting (vertex shader)

```
void C6E4v_litKeyFrame(float3 positionA : POSITION,  
                       float3 normalA   : NORMAL,  
                       float3 positionB : TEXCOORD1,  
                       float3 normalB   : TEXCOORD2,  
                       float2 texCoord  : TEXCOORD0,  
                       out float4 oPosition : POSITION,  
                       out float2 oTexCoord : TEXCOORD0,  
                       out float4 color   : COLOR,  
                       uniform float     keyFrameBlend,  
                       uniform Light     light,  
                       uniform float4x4  modelViewProj)  
{  
    float3 position = lerp(positionA, positionB,  
                           keyFrameBlend);  
    float3 blendNormal = lerp(normalA, normalB,  
                              keyFrameBlend);  
    float3 normal = normalize(blendNormal);  
    oPosition = mul(modelViewProj, float4(position, 1));  
    oTexCoord = texCoord;  
    color = computeLighting(light, position, normal);  
}
```

# Compute lighting

```
float4 computeLighting(Light light,
                      float3 position, // In object space
                      float3 normal) // In object space
{
    float3 lightDirection = light.lightPosition - position;
    float3 lightDirNorm = normalize(lightDirection);
    float3 eyeDirection = light.eyePosition - position;
    float3 eyeDirNorm = normalize(eyeDirection);
    float3 halfAngle = normalize(lightDirNorm + eyeDirNorm);
    float diffuse = max(0, dot(lightDirNorm, normal));
    float specular = pow(max(0, dot(halfAngle, normal)),
                        light.specularExponent);
    return light.lightColor * (light.ambient +
                              diffuse + specular);
}
```

# Skinning, part 1 (vertex shader)

```
void C6E5v_skin4m(float3 position      : POSITION,
                 float3 normal        : NORMAL,
                 float2 texCoord      : TEXCOORD0,
                 float4 weight        : TEXCOORD1,
                 float4 matrixIndex   : TEXCOORD2,

                 out float4 oPosition : POSITION,
                 out float2 oTexCoord : TEXCOORD0,
                 out float4 color      : COLOR,

uniform Light light,
uniform float4  boneMatrix[72], // 24 matrices
uniform float4x4 modelViewProj)
{
    ...
}
```

# Skinning, part 2 (vertex shader)

```
...  
  
float3 netPosition = 0, netNormal = 0;  
  
for (int i=0; i<4; i++) {  
    float index = matrixIndex[i];  
    float3x4 model = float3x4(boneMatrix[index+0],  
                               boneMatrix[index+1],  
                               boneMatrix[index+2]);  
    float3 bonePosition = mul(model, float4(position, 1));  
    // Assume no scaling in matrix, just rotate & translate  
    float3x3 rotate = float3x3(model[0].xyz,  
                                model[1].xyz,  
                                model[2].xyz);  
    float3 boneNormal = mul(rotate, normal);  
    netPosition += weight[i] * bonePosition;  
    netNormal    += weight[i] * boneNormal;  
}  
  
...
```



# Skinning, part 3 (vertex shader)

...

```
netNormal = normalize(netNormal);  
oPosition = mul(modelViewProj, float4(netPosition, 1));  
oTexCoord = texCoord;  
color = computeLighting(light, netPosition, netNormal);  
}
```