CPSC453

2D to 3D

Vertex Info and Lighting

UNIVERSITY OF CALGARY

2D → 3D

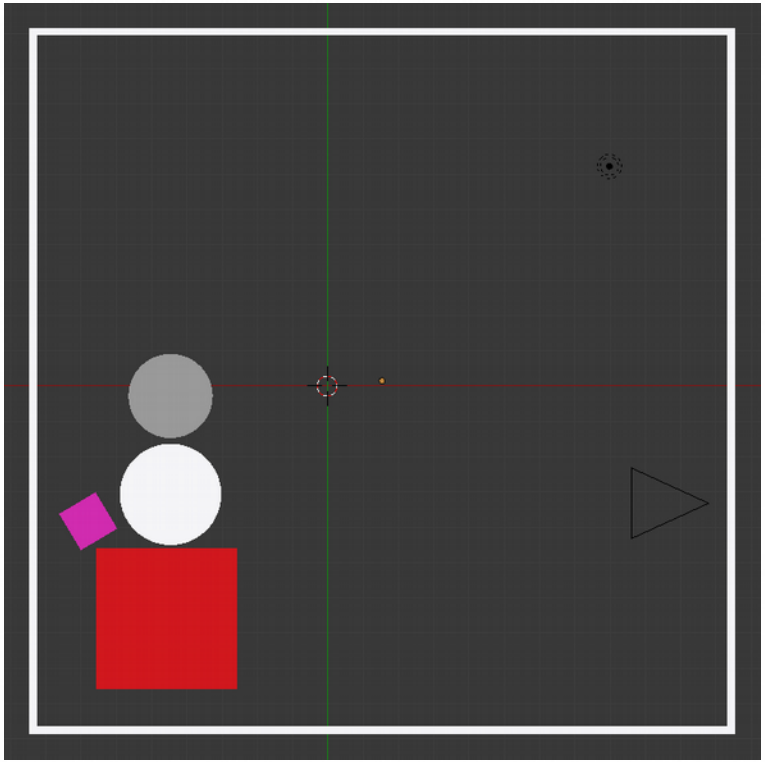
UNIVERSITY OF CALGARY

2D ≈ 3D

- More Data
- More Potential Realism

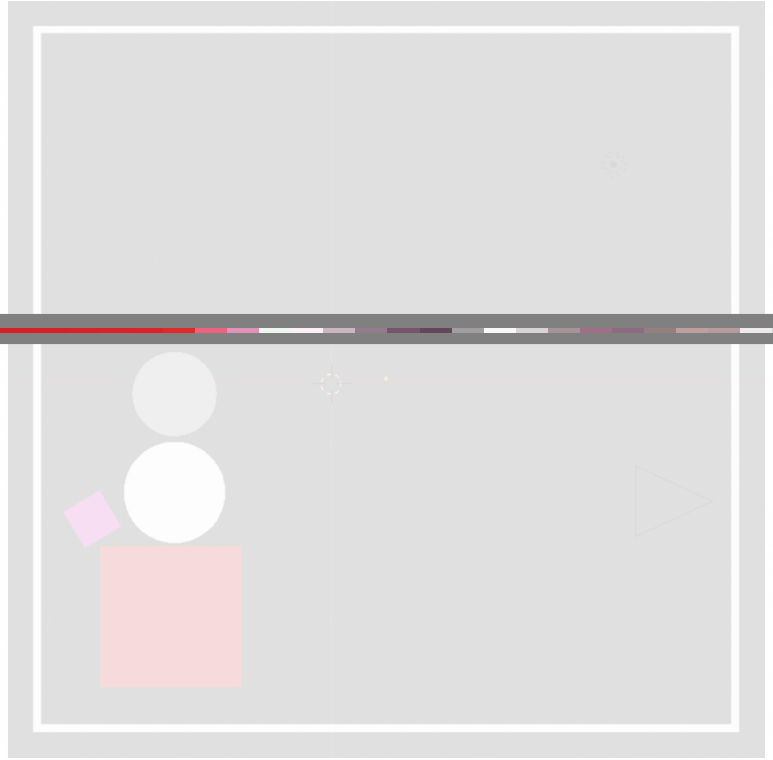
UNIVERSITY OF CALGARY

More Realism?



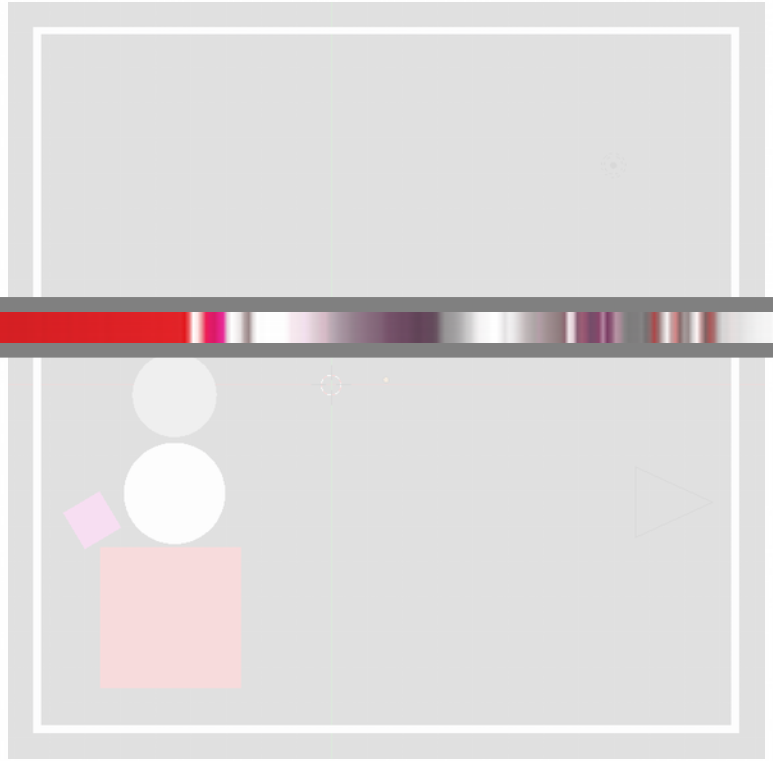
UNIVERSITY OF CALGARY

More Realism?



UNIVERSITY OF CALGARY

More Realism?



UNIVERSITY OF CALGARY

More Realism

Local Illumination: *Phong illumination model*

Phong Illumination

$$I_r = k_a I_a + k_d I_d + k_s I_s$$

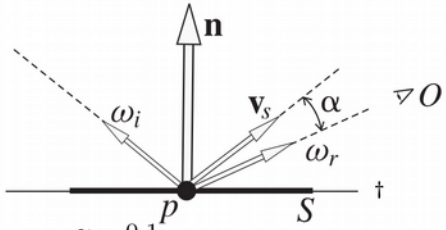
reflected towards observer — ambient — diffuse — specular-diffuse

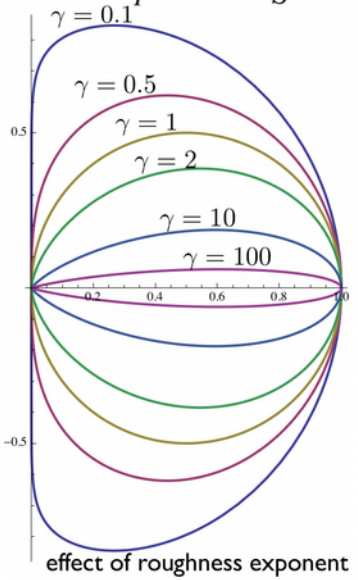
I_a is constant

$$I_d = I_i \langle \omega_i, \mathbf{n} \rangle = I_i \cos(\theta_i)$$

$$I_s = I_i \langle \mathbf{v}_s, \omega_r \rangle^{\gamma} = I_i \cos^{\gamma}(\alpha)$$

$$I_r = k_a I_a + I_i \left(k_d \cos(\theta_i) + k_s \cos^{\gamma}(\alpha) \right)$$





effect of roughness exponent γ

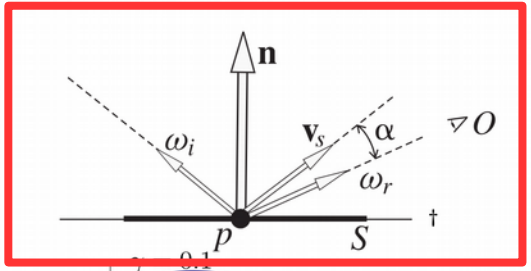
More Realism

Local Illumination: *Phong illumination model*

Phong Illumination

$$I_r = k_a I_a + k_d I_d + k_s I_s$$

reflected towards observer — ambient — diffuse — specular-diffuse

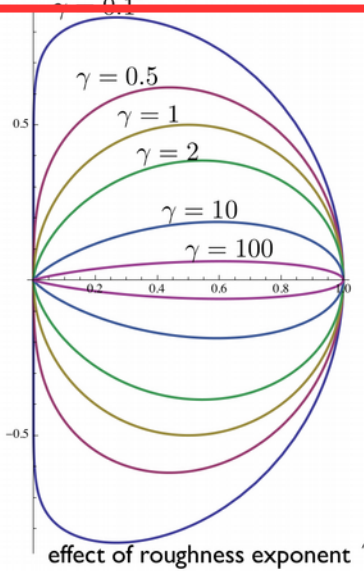


I_a is constant

$$I_d = I_i \langle \omega_i, \mathbf{n} \rangle = I_i \cos(\theta_i)$$

$$I_s = I_i \langle \mathbf{v}_s, \omega_r \rangle^{\gamma} = I_i \cos^{\gamma}(\alpha)$$

$$I_r = k_a I_a + I_i \left(k_d \cos(\theta_i) + k_s \cos^{\gamma}(\alpha) \right)$$



effect of roughness exponent γ

Things that Vary

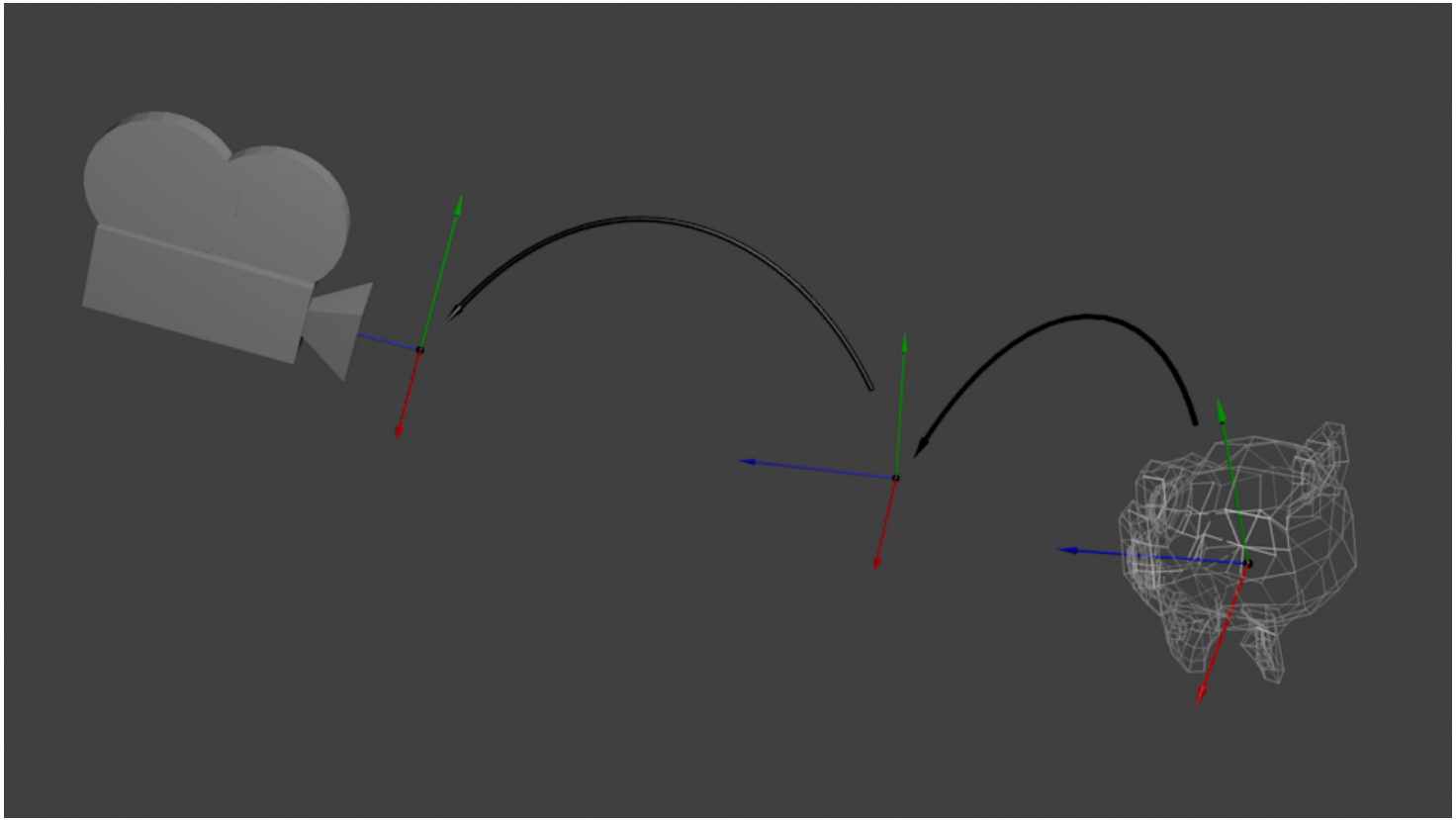
Eye Vector
Normal Vector
Object Colour

Things that Don’t

Ambient Level
Diffuse Light Vector(s)
Diffuse Light Colour(s)
Specular Power(s)
Specular Colour(s)

UNIVERSITY OF CALGARY

OpenGL Transforms



http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/

Eye Vector

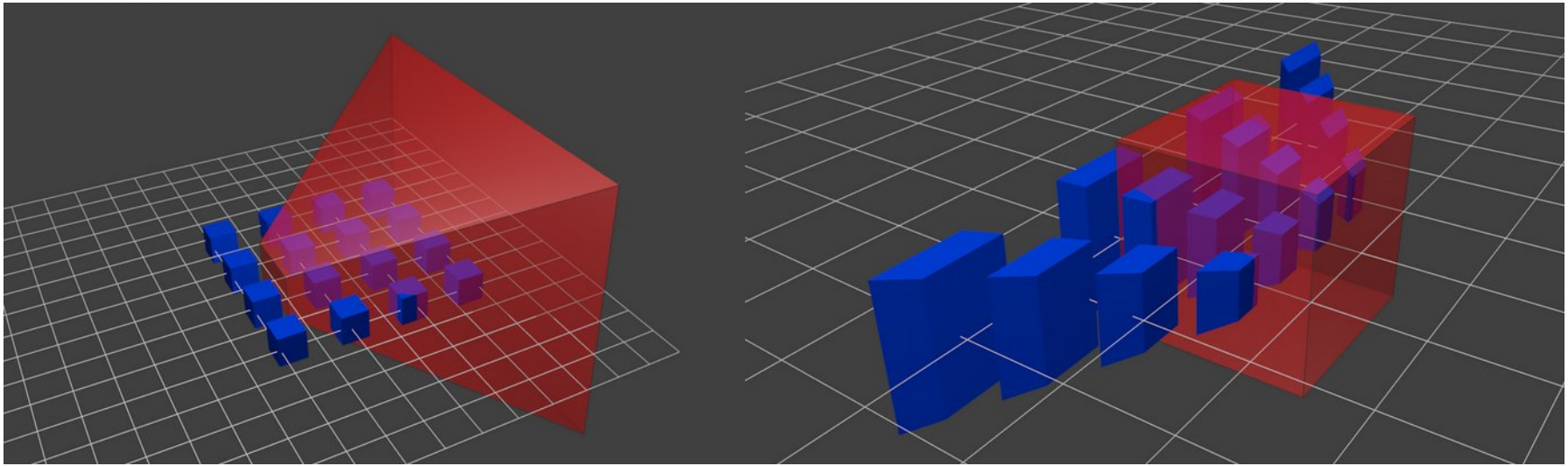
Model to World ✔

World to View ✔

View to Perspective X

UNIVERSITY OF CALGARY

OpenGL Transforms



http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/

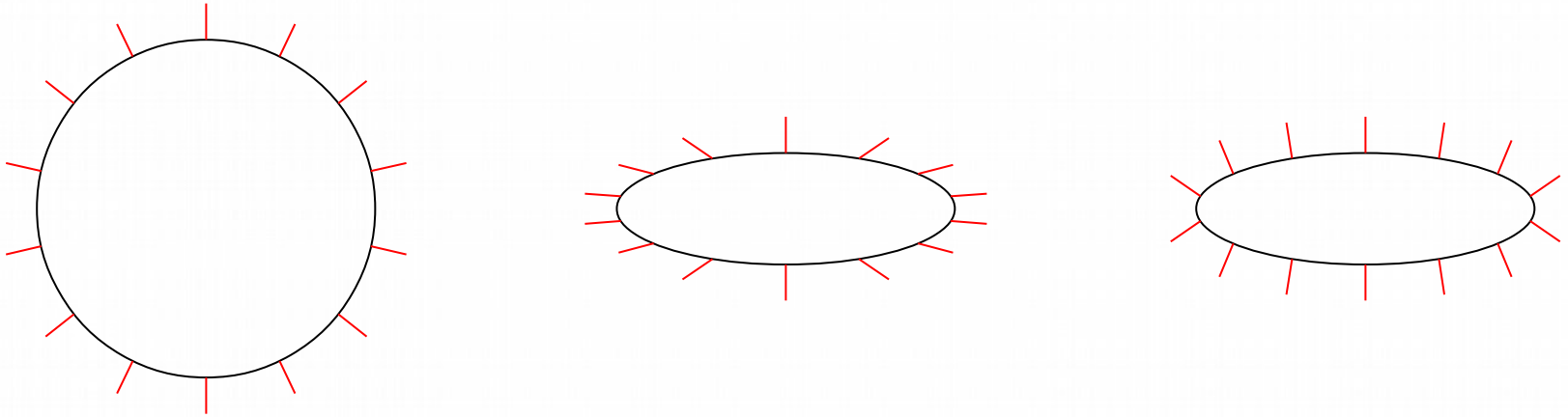
Eye Vector

Model to World	✔
World to View	✔
View to Perspective	X

Normal Vector

Model to World	X
World to View	X
View to Perspective	X

Dealing with Normals



https://paroj.github.io/gltut/Illumination/Tut09%20Normal%20Transformation.html

Dealing with Normals

$$MV_{\text{normal}} = \left(MV^{-1}\right)^{T}$$

https://paroj.github.io/gltut/Illumination/Tut09%20Normal%20Transformation.html

Eye Vector

Model to World	✔
World to View	✔
View to Perspective	X

Normal Vector

Model to World	✔*
World to View	✔*
View to Perspective	X

* Needs a special matrix

Vertex Shader

```
uniform mat4 modelview;              // model → visual location
uniform mat4 perspective;            // visual → perspective warp
uniform mat3 MVinvTrans;             // inverse transpose of MV

in vec3 position;
In vec3 normal;

out vec4 fragPosition;                    // or to tesselation/geometry
out vec3 fragEye;
out vec3 fragNormal;


void main() {

    fragEye = modelview * -vec4(position, 1.0);
    fragNormal = MVinvTrans * normal;
    fragPosition = perspective * -fragEye;

    }
```

Fragment Shader

```
uniform vec3 lightAmbient;
uniform vec3 lightDir;          // normalized, and in view space!!
uniform vec3 diffuseColour;     // almost certainly vec3(1,1,1)
uniform float specPower;
uniform vec3 specColour;

in vec4 fragPosition;
in vec3 fragEye;
in vec3 fragNormal;

out vec4 colour;

void main() {

    vec3 light;
    // normalize the eye and normal vectors
    // diffuse intensity = dot product of normal and light vector
    // clamp diffuse intensity to [0:1]
    // half angle = average of lightDir and eye vector**
    // specular weight = dot product of half angle and normal
    // clamp specular weight to [0:1]
    // light = lightAmbient + (diffuse intensity * diffuseColour)
    // light += specColour * (specular weight raised to specPower)
    // colour = texture2D() * light
    // clamp “colour” to [0:1]
    // gamma correct “colour”


    }

// ** This is actually Blinn-Phong, rather than Phong Classic.
```

Vertex Shader

```
uniform mat4 modelview;          // model → visual location
uniform mat4 perspective;        // visual → perspective warp
uniform mat3 MVinvTrans;         // inverse transpose of MV

in vec3 position;                // two inputs?!
in vec3 normal;

out vec4 fragPosition;               // or to tesselation/geometry
out vec3 fragEye;
out vec3 fragNormal;

void main() {

    fragEye = -modelview * vec4(position, 1.0);
    fragNormal = MVinvTrans * normal;
    fragPosition = perspective * -eye;

    }
```

UNIVERSITY OF CALGARY

Communication

1) Indirect / Direct
2) glUniform()
3) Textures
4) Uniform Buffer Objects
5) Shader Storage Buffer Objects

UNIVERSITY OF CALGARY

Direct

Boilerplate's Way

```
// boilerplate.cpp

  void addBuffer(string name, int index, vector<float> buffer) {

    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

Boilerplate's Way

```
// boilerplate.cpp
  void addBuffer(string name, int index, vector<float> buffer) {
    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

Boilerplate's Way

```
// boilerplate.cpp

  void addBuffer(string name, int index, vector<float> buffer) {

    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

Boilerplate's Way

```
// boilerplate.cpp
  void addBuffer(string name, int index, vector<float> buffer) {
    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

UNIVERSITY OF CALGARY

Boilerplate's Way

```
// boilerplate.cpp

  void addBuffer(string name, int index, vector<float> buffer) {

    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

Boilerplate's Way

```
// boilerplate.cpp

  void addBuffer(string name, int index, vector<float> buffer) {

    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

UNIVERSITY OF CALGARY

Boilerplate's Way

```
// boilerplate.cpp

  void addBuffer(string name, int index, vector<float> buffer) {

    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

Boilerplate's Way

```
// boilerplate.cpp
  void addBuffer(string name, int index, vector<float> buffer) {
    GLuint buffer_id;
    glBindVertexArray(id);

    glGenBuffers(1, &buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, buffer_id);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    buffers[name]=buffer_id;
    indices[name]=index;

    int components=buffer.size()/count;
    glVertexAttribPointer(index, components, GL_FLOAT, GL_FALSE, 0, 0);
    glEnableVertexAttribArray(index);

    // unset states
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
  }

  void updateBuffer(string name, vector<float> buffer) {
    glBindBuffer(GL_ARRAY_BUFFER, buffers[name]);
    glBufferData(GL_ARRAY_BUFFER, buffer.size()*sizeof(float), buffer.data(),
GL_STATIC_DRAW);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
  }
```

UNIVERSITY OF CALGARY

Boilerplate's Way

<px,py,pz> , <px,py,pz> , <px,py,pz> , <px,py,pz> , ….

<nx,ny,nz> , <nx,ny,nz> , <nx,ny,nz> , <nx,ny,nz> , ….

<s,t> , <s,t> , <s,t> , <s,t> , ….

UNIVERSITY OF CALGARY

Interleaved Values

<px,py,pz> , <nx,ny,nz> , <s,t> , <px,py,pz> , <nx,ny,nz> , <s,t> , ….

Interleaved Values

```
// CPU
int stride = (3 + 3 + 2) * sizeof(float);    // how many bytes per vertex?

glVertexAttribPointer( 0, 3, GL_FLOAT, false, stride, (GLvoid*)(0) );
glVertexAttribPointer( 1, 3, GL_FLOAT, false, stride, (GLvoid*)(0 + 3*sizeof(float)) );
glVertexAttribPointer( 2, 2, GL_FLOAT, false, stride, (GLvoid*)(0 + 6*sizeof(float)) );

for (uint it = 0; it < 3; it++)
     glEnableVertexAttribArray(it);


// vertex shader
layout(location = 0) in vec3 position;
layout(location = 1) in vec3 normal;
Layout(location = 2) in vec2 texture;

// source: http://docs.gl/gl4/glVertexAttribPointer
// source: http://antongerdelan.net/opengl/vertexbuffers.html
```

Interleaved Values

```
// CPU
int stride = (3 + 3 + 2) * sizeof(float);    // how many bytes per vertex?

glVertexAttribPointer( 0, 3, GL_FLOAT, false, stride, (GLvoid*)(0) );
glVertexAttribPointer( 1, 3, GL_FLOAT, false, stride, (GLvoid*)(0 + 3*sizeof(float)) );
glVertexAttribPointer( 2, 2, GL_FLOAT, false, stride, (GLvoid*)(0 + 6*sizeof(float)) );

for (uint it = 0; it < 3; it++)
     glEnableVertexAttribArray(it);


// vertex shader
layout(location = 0) in vec3 position;
layout(location = 1) in vec3 normal;
Layout(location = 2) in vec2 texture;

// source: http://docs.gl/gl4/glVertexAttribPointer
// source: http://antongerdelan.net/opengl/vertexbuffers.html
```

glVertexArrayPointer()

- Best documented
- Considered obsolete

glVertexAttribFormat()

- Better approach
- Not much documentation

glVertexArrayPointer()

- Best documented
- Considered obsolete

glVertexAttribFormat()

- Better approach
- Not much documentation

https://www.khronos.org/opengl/wiki/Vertex_Specification#Separate_attribute_format

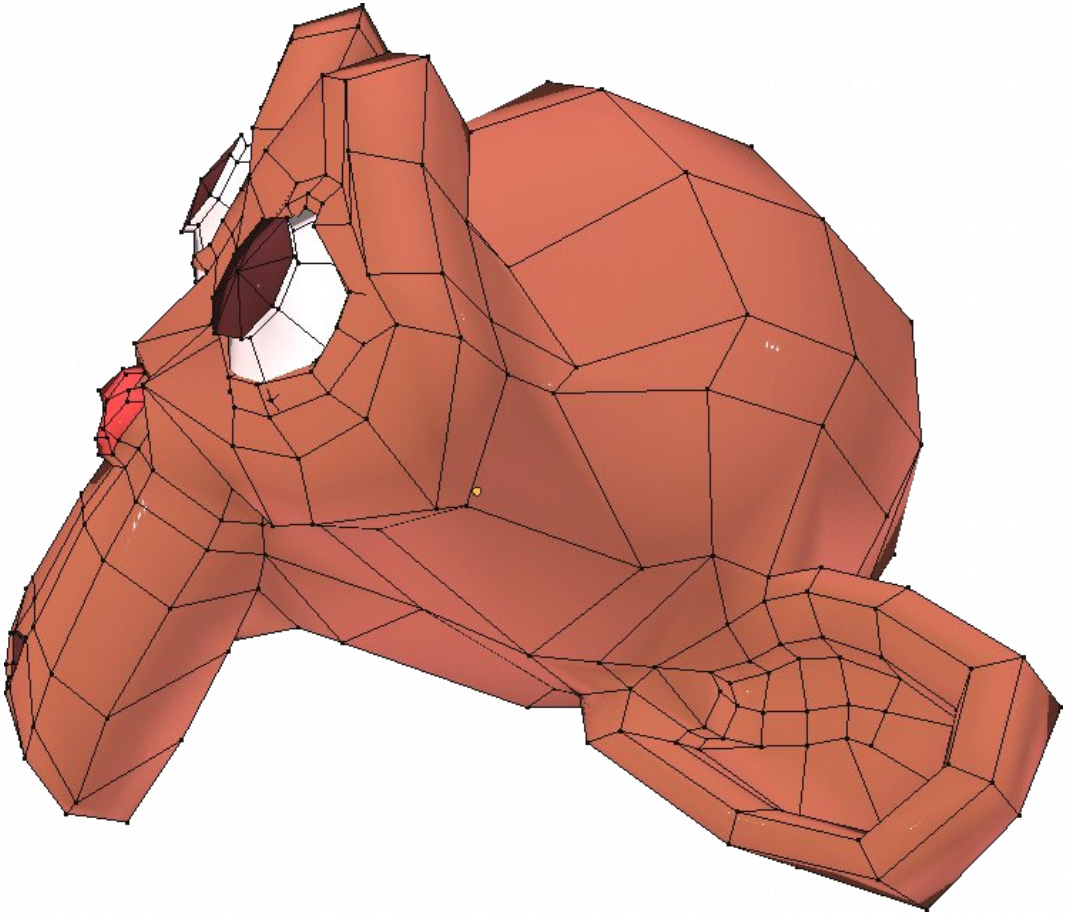
UNIVERSITY OF CALGARY

OBJ files

“demo” 6

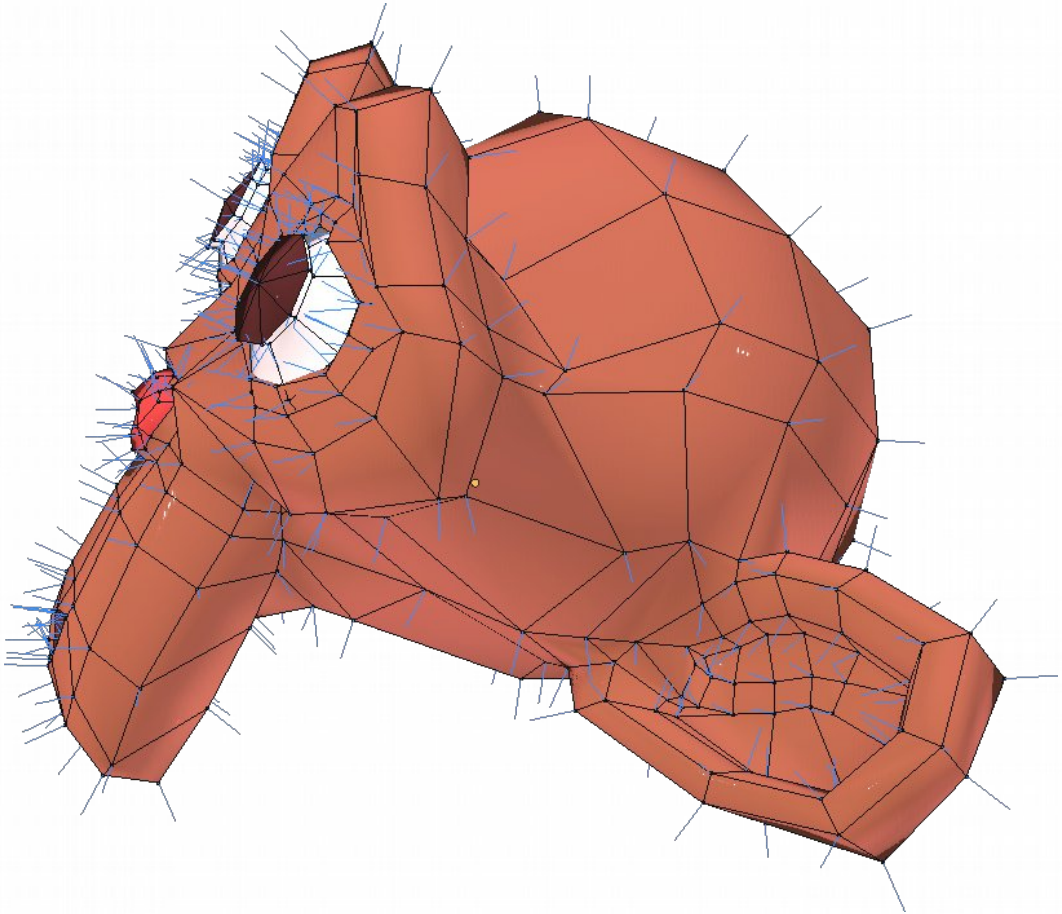
UNIVERSITY OF CALGARY

Representing Objects



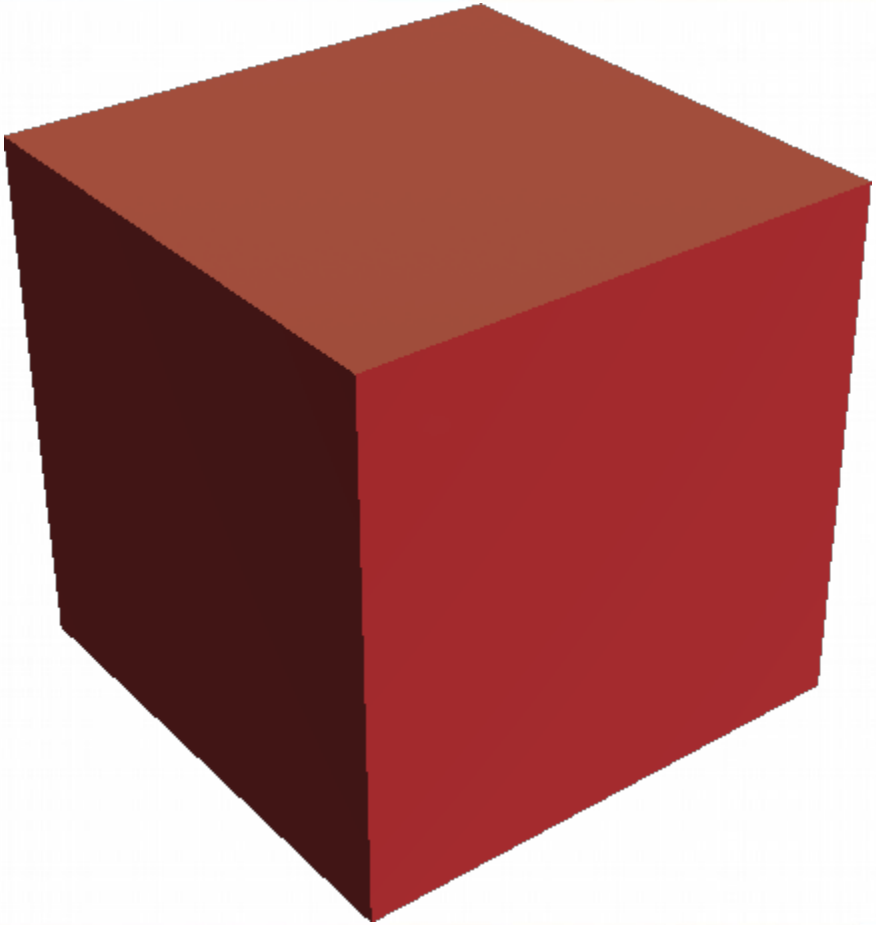
UNIVERSITY OF CALGARY

Representing Objects



UNIVERSITY OF CALGARY

Representing Objects



UNIVERSITY OF CALGARY

Representing Objects

UNIVERSITY OF CALGARY

Representing Objects

UNIVERSITY OF CALGARY

OBJ format

```
# Blender v2.78 (sub 0) OBJ File: ''
# www.blender.org
mtllib timmy_cup.mtl
o timmy_cup
v 0.025373 0.078107 0.027071
v 0.026879 0.058619 0.023547
V 0.027913 0.078107 0.024452
# …
vt 0.4494 0.9391
vt 0.4530 0.9388
vt 0.4530 0.9390
# …
vn -0.2363 0.3306 0.9137
vn -0.2430 0.2408 0.9397
vn -0.2477 0.1474 0.9576
# …
usemtl plastic.top
s off
f 2210/1/1 2211/2/1 2212/3/1
f 2213/4/2 2214/5/2 2211/2/2
f 2214/5/3 2215/6/3 2216/7/3
# …
```

OBJ format

```
# Blender v2.78 (sub 0) OBJ File: ''
# www.blender.org
mtllib timmy_cup.mtl
o timmy_cup
v 0.025373 0.078107 0.027071
v 0.026879 0.058619 0.023547
V 0.027913 0.078107 0.024452
# …
vt 0.4494 0.9391
vt 0.4530 0.9388
vt 0.4530 0.9390
# …
vn -0.2363 0.3306 0.9137
vn -0.2430 0.2408 0.9397
vn -0.2477 0.1474 0.9576
# …
usemtl plastic.top
s off
f 2210/1/1 2211/2/1 2212/3/1
f 2213/4/2 2214/5/2 2211/2/2
f 2214/5/3 2215/6/3 2216/7/3
# …
```