SIGGRAPH 2014 Studio Course Submission

Developing a 3D Model Viewer for iOS using COLLADA and OpenGL ES

Ricardo Rendon Cepeda
Idean / RayWenderlich.com
ricardo@idean.com

https://github.com/ricardo-rendoncepeda/RRCModelViewer

Schedule

09:00am - 12:15pm

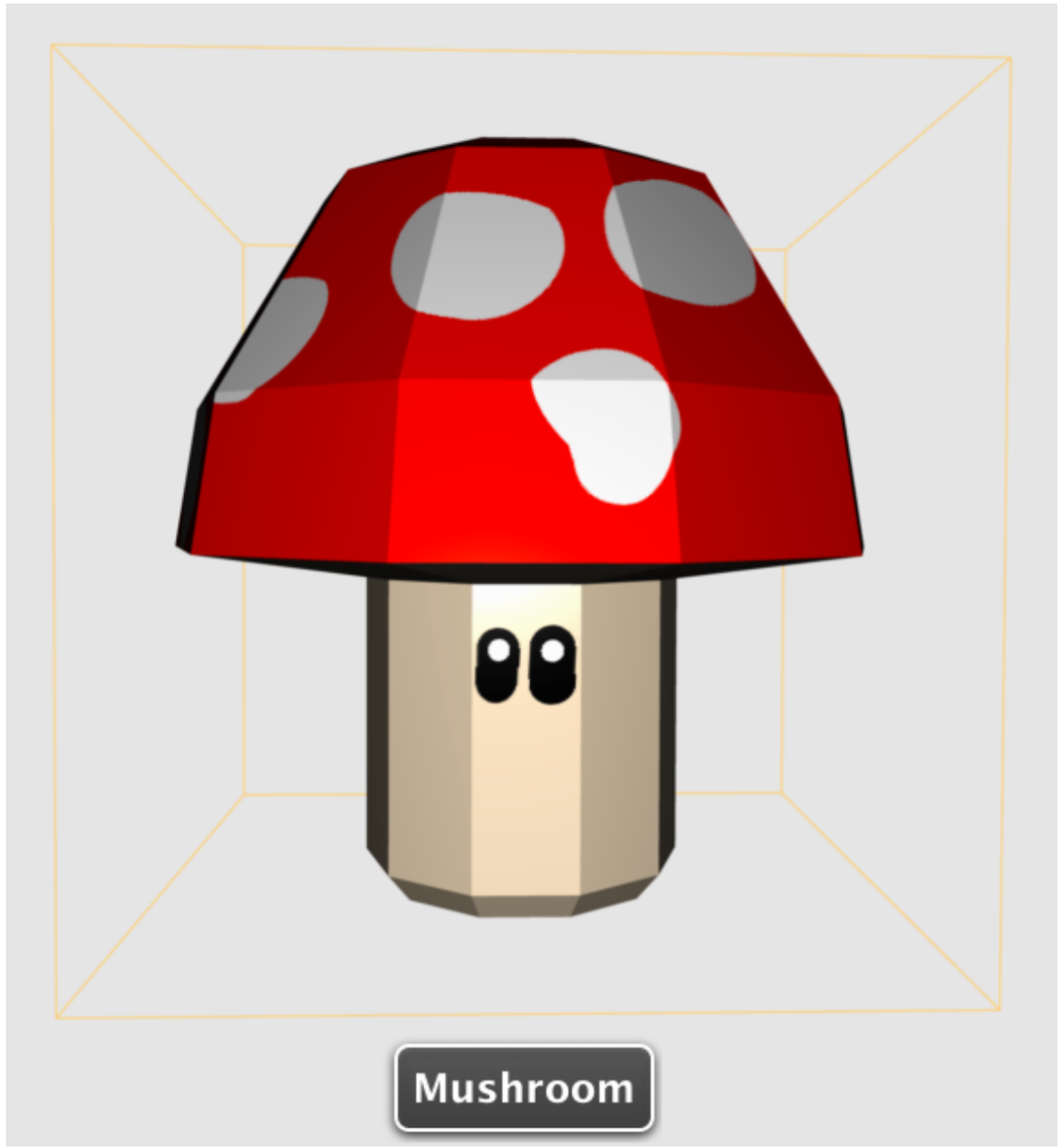
- **09:00am** Introduction / Welcome
- **09:15am** COLLADA
- **09:45am** OpenGL ES
- **10:15am** GLKit and GLKBaseEffect
- **10:45am** -BREAK-
- **11:00am** Shaders and GLSL
- **11:30am** User Interface and Gesture Recognizers
- **12:00pm** Conclusion / Q&A

COLLADA (.dae)

Digital **A**sset **E**xchange

Compatible with…

- 3D: Maya, Blender
- 2D: Preview, Photoshop
- Development: Xcode / SceneKit



COLLADA (.xml)

E**x**tensible **M**arkup **L**anguage

- geometry
- mesh
- source
- float_array
- accessor
- input / vertices
- polylist / triangles

```
<geometry id="MushroomGeometry-mesh" name="MushroomGeometry">
  <mesh>
    <source id="MushroomGeometry-mesh-positions">
      <float_array id="MushroomGeometry-mesh-positions-array" count="216">0 0 1.5 0 1.177
          0.9522407 -1.5 0.5874974 0.8086208 1.5 1.119426 0.3637235 -1.5 0.9505907 0.3088
          0.9505906 -0.3088656 1.5 0.6918432 -0.9522405 -1.5 0.5874972 -0.8086207 1.5 0 -
          -0.6918434 -0.9522404 -1.5 -0.5874975 -0.8086206 1.5 -1.119426 -0.3637236 -1.5
          0.3637237 -1.5 -0.9505906 0.3088658 1.5 -0.6918429 0.952241 -1.5 -0.587497 0.80
          1.213526 1.294463 1.426585 0.4635255 1.294463 1.426585 -0.4635255 1.294463 0.88
          -1.5 1.294463 -0.8816781 -1.213525 1.294463 -1.426585 -0.4635257 1.294463 -1.42
          1.213526 1.294463 1.536202 2.114401 -1.289925 2.485628 0.8076294 -1.289925 2.48
          -2.114401 -1.289925 -2.37333e-7 -2.613544 -1.289925 -1.536203 -2.114401 -1.2899
          -2.485628 0.8076297 -1.289925 -1.536202 2.114402 -1.289925 0 2.613544 -1.289925
          0.7567244 -0.2689725 2.328958 -0.7567246 -0.2689725 1.439375 -1.98113 -0.268972
          -1.439376 -1.98113 -0.2689725 -2.328958 -0.7567248 -0.2689725 -2.328958 0.75672
          -0.2689725 0 2.448812 -0.2689725 0 1.177034 -3.5844 0.6918433 0.9522407 -3.5844
          -0.3637235 -3.5844 0.6918432 -0.9522405 -3.5844 0 -1.177034 -3.5844 -0.6918434
          -0.3637236 -3.5844 -1.119426 0.3637237 -3.5844 -0.6918429 0.952241 -3.5844 0 0
          0.5872793 0.8083207 -3.788765 0.9502379 0.308751 -3.788765 0.9502378 -0.308751
          -3.788765 0 -0.9991392 -3.788765 -0.5872794 -0.8083205 -3.788765 -0.9502378 -0.
          -3.788765 -0.587279 0.8083209 -3.788765</float_array>
<technique_common>
  <accessor source="#MushroomGeometry-mesh-positions-array" count="72" stride="3">
    <param name="X" type="float"/>
    <param name="Y" type="float"/>
    <param name="Z" type="float"/>
  </accessor>
</technique_common>
```

OpenGL ES Rendering Cycle

- **glEnableVertexAttribArray**
  Enable a generic vertex attribute array (e.g. positions data).

- **glVertexAttribPointer**
  Define an array of generic vertex attribute data (e.g. float array).

- **glDrawArrays**
  Render primitives from array data (e.g. draw triangles).

```
glEnableVertexAttribArray( GLuint index );

glVertexAttribPointer( GLuint indx,
                       GLint size,
                       GLenum type,
                       GLboolean normalized,
                       GLsizei stride,
                       const GLvoid *ptr );

glDrawArrays( GLenum mode,
              GLint first,
              GLsizei count );
```

OpenGL ES Object Model

1. Define a generic, re-usable OpenGL ES Object Model.

2. Parse relevant COLLADA elements by tags.

3. Load COLLADA content into memory.

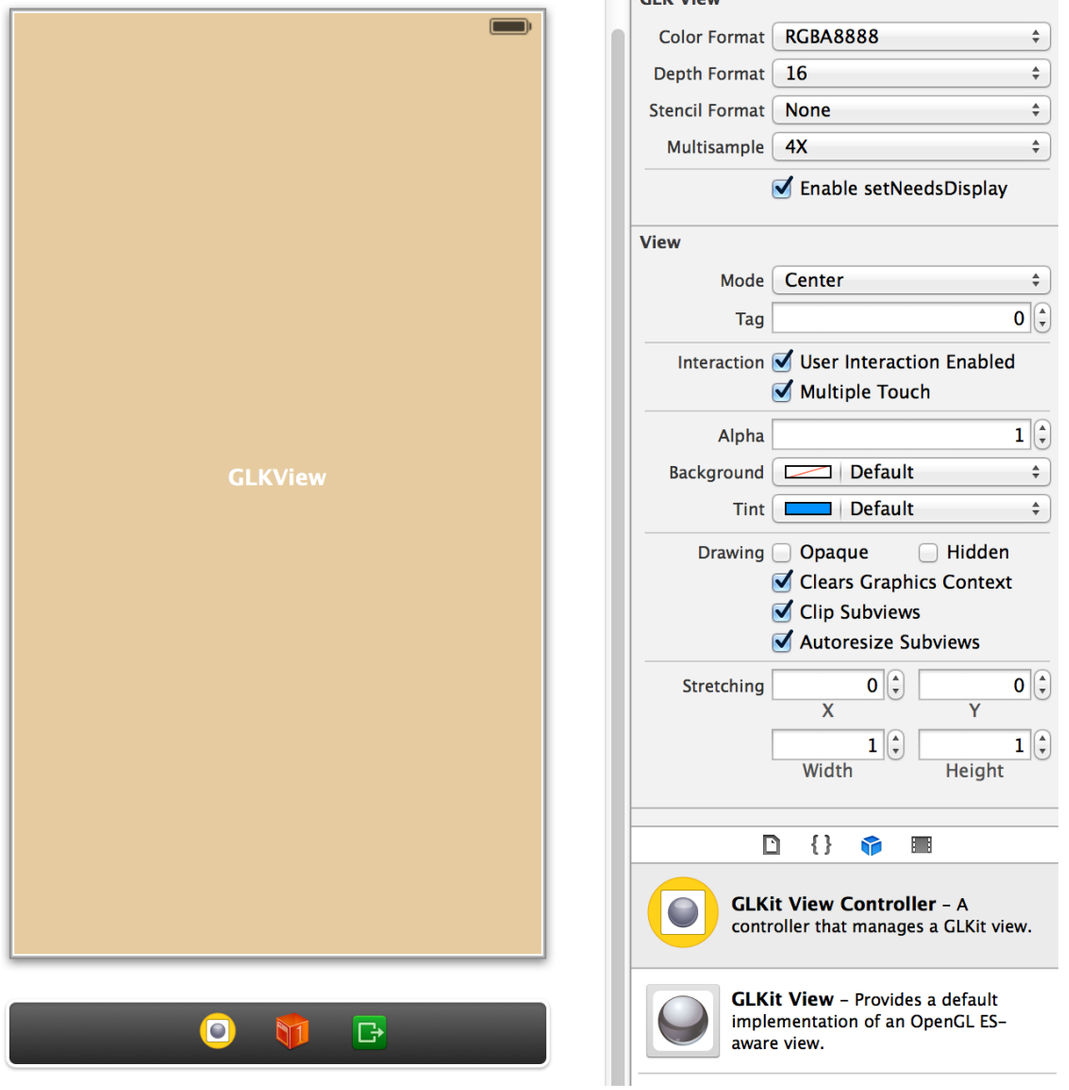
4. Convert COLLADA content into OpenGL ES arrays/data suitable for rendering.

```
#import "RRCColladaObject.h"

@interface RRCOpenglesModel : NSObject

@property (assign, nonatomic, readonly) int count;
@property (assign, nonatomic, readonly) float* positions;
@property (assign, nonatomic, readonly) float* normals;
@property (assign, nonatomic, readonly) float* texels;

- (instancetype)initWithCollada:(RRCColladaObject*)collada;

@end
```

GLKit

Benefits of using GLKit

- Easy setup with GLKView
- 1-step multisampling
- Automatic buffer handling
- *update* and *glkView:drawInRect:*
- Robust vector/matrix math library



GLKBaseEffect

- Very easy setup
- Mimics OpenGL ES 1.1 (fixed-function pipeline)
- No shader support (programmable pipeline)
- No points!
- Supports position, normal, color, and texture coordinate vertex attributes

```
- (void)createEffect
{
    // Initialize
    self.effect = [[GLKBaseEffect alloc] init];

    // Texture
    NSDictionary* options = @{GLKTextureLoaderOriginBottomLeft:@YES};
    NSError* error;
    NSString* resource = [NSString stringWithFormat:@"Models/%@Texture", kRRCModel];
    NSString* path = [[NSBundle mainBundle] pathForResource:resource ofType:@".png"];
    GLKTextureInfo* texture = [GLKTextureLoader textureWithContentsOfFile:path options:options error:&error];

    if(texture == nil)
        NSLog(@"%@:- Error loading texture: %@", [self class], [error localizedDescription]);

    self.effect.texture2d0.name = texture.name;
    self.effect.texture2d0.enabled = true;

    // Light
    self.effect.light0.enabled = GL_TRUE;
    self.effect.light0.position = GLKVector4Make(0.25f, 0.75f, 0.75f, 1.0f);
    self.effect.lightingType = GLKLightingTypePerPixel;

    // Material
    self.effect.material.diffuseColor = GLKVector4Make(0.75f, 0.75f, 0.75f, 1.0f);
    self.effect.material.specularColor = GLKVector4Make(0.25f, 0.25f, 0.25f, 1.0f);
}
```

Shaders (GLSL / GPU)

- Open**GL S**hading **L**anguage
  C-like language
  Compiled, linked, and run on the GPU

- **Vertex shader**
  Called once per *vertex*
  Computes *gl_Position*

- **Fragment shader**
  Called once per *fragment*
  Computes *gl_FragColor*

```
// Vertex Shader

static char* const BlinnPhongVSH = STRINGIFY
(
 // Attributes
 attribute vec3 aPosition;
 attribute vec3 aNormal;
 attribute vec2 aTexel;

 // Uniforms
 uniform mat4 uProjectionMatrix;
 uniform mat4 uModelViewMatrix;
 uniform mat3 uNormalMatrix;

 // Varying
 varying vec3 vNormal;
 varying vec2 vTexel;

 void main(void)
 {
     vNormal = uNormalMatrix * aNormal;
     vTexel = aTexel;

     gl_Position = uProjectionMatrix * uModelViewMatrix * vec4(aPosition, 1.0);
 }
);
```

Shaders (Objective-C / CPU)

- Program handle
  Vertex shader + Fragment shader

- Attribute handles
  Per-vertex variables (e.g. positions)

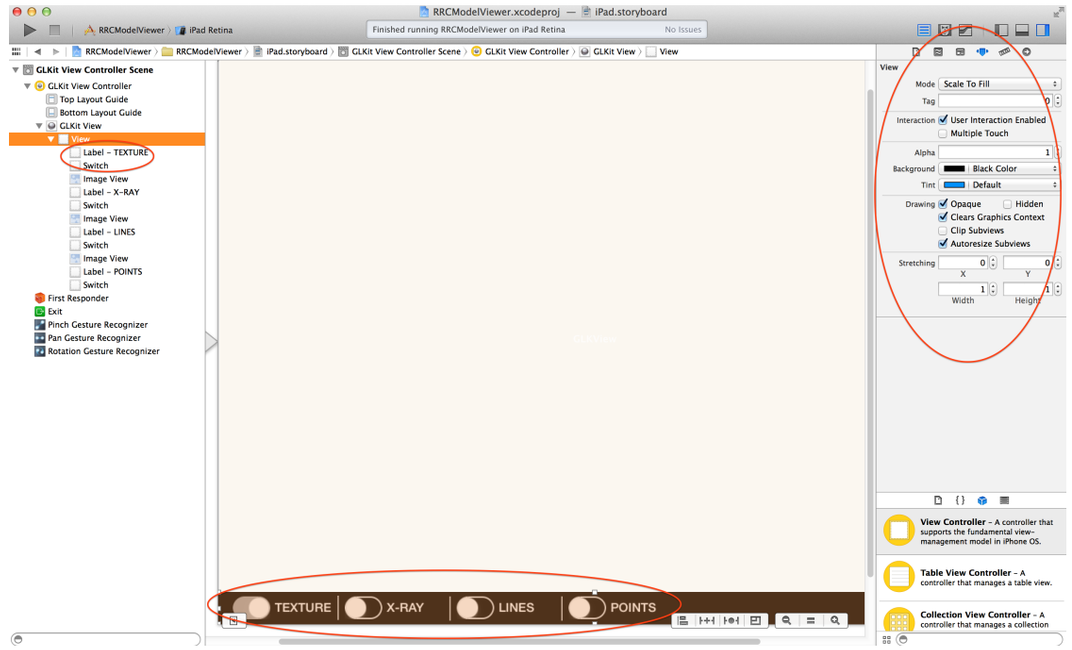
- Uniform handles
  Per-frame variables (e.g. projection matrix)

```objc
// Shaders
#include "BlinnPhong.vsh"
#include "BlinnPhong.fsh"

@implementation RRCShaderBlinnPhong

#pragma mark - init
- (instancetype)init
{
    if(self = [super initWithVertexShader:BlinnPhongVSH fragmentShader:BlinnPhongFSH])
    {
        // Attributes
        _aNormal = glGetAttribLocation(self.program, "aNormal");
        _aTexel = glGetAttribLocation(self.program, "aTexel");

        // Uniforms
        _uNormalMatrix = glGetUniformLocation(self.program, "uNormalMatrix");
        _uTexture = glGetUniformLocation(self.program, "uTexture");
        _uSwitchTexture = glGetUniformLocation(self.program, "uSwitchTexture");
        _uSwitchXRay = glGetUniformLocation(self.program, "uSwitchXRay");
    }
    return self;
}
```

User Interface

Drag, Drop, Customize, and Connect!

- UIView
- UISwitch
- UILabel
- UIImageView



Gesture Recognizers

- **UIPinchGestureRecognizer**
  Scale XYZ

- **UIPanGestureRecognizer**
  Translate XY (1 finger)
  Rotate XY (2 fingers)

- **UIRotationGestureRecognizer**
  Rotate Z