Introduction to Computer Graphics with WebGL

Rongkai Guo
Assistant Professor at Computer Game Design program
Kennesaw State University
Overview

• These lectures are for a senior/graduate elective for computer science and engineering majors (and others with good programming skills)

• The course is based on a modern approach using programmable shaders in the new textbook: Ed Angel and Dave Shreiner, Interactive Computer Graphics, A Top-down Approach with WebGL (Seventh Edition), Addison-Wesley

• These lectures cover Chapters 1-7 in detail and survey Chapters 8-12
Lecture 1

• 1.1: Introduction
• 1.2: Detailed Outline and Examples
• 1.3: Example Code in JS
• 1.4: What is Computer Graphics?
• 1.5: Image Formation

• Reading: Chapter 1
• Exercises: Run some examples on your browser
1.1

• Course Overview
• Required Background
• References
Contact Information

rguo@Kennesaw.edu

http://ksuweb.kennesaw.edu/~rguo/

http://ksuweb.kennesaw.edu/~rguo/2017_Summer/CS4722-7327/
Objectives

• Broad introduction to Computer Graphics
  • Software
  • Hardware
  • Applications

• Top-down approach

• Shader-Based WebGL
  • Integrates with HTML5
  • Code runs in latest browsers
Prerequisites

- Good programming skills in C (or C++)
- Basic Data Structures
  - Linked lists
  - Arrays
- Geometry
- Simple Linear Algebra
Requirements

• 3 Assigned Projects
  • Simple
  • Interactive
  • 3D
• Term Project
  • You pick
• See
  http://ksuweb.kennesaw.edu/~rguo/2017_Summer-CS472_2-7327/

for assignments, projects and other info
Why is this course different?

• Shader-based
  • Most computer graphics use OpenGL but still use fixed-function pipeline
  • does not require shaders
  • Does not make use of the full capabilities of the graphics processing unit (GPU)

• Web
  • With HTML5, WebGL runs in the latest browsers
  • makes use of local hardware
  • no system dependencies
References

• Interactive Computer Graphics (7th Edition)
• The OpenGL Programmer’s Guide (the Redbook) 8th Edition
• OpenGL ES 2.0 Programming Guide
• WebGL Programming Guide
• WebGL Beginner’s Guide
• WebGL: Up and Running
• JavaScript: The Definitive Guide
Web Resources

- www.opengl.org
- get.webgl.org
- www.kronos.org/webgl
- www.chromeexperiments.com/webgl
- learningwebgl.com
1.2 Course Outline
Outline: Part 1

• Introduction
• Text: Chapter 1
• Lecture 1
  • What is Computer Graphics?
  • Applications Areas
  • History
  • Image formation
  • Basic Architecture
Outline: Part 2

• Basic WebGL Graphics
• Text: Chapter 2
• Lectures 2-4
  • Architecture
  • JavaScript
  • Web execution
  • Simple programs in two and three dimensions
  • Basic shaders and GLSL
Outline: Part 3

- Interaction
- Text: Chapter 3
- Lecture 4
  - Client-Server Model
  - Event-driven programs
  - Event Listeners
  - Menus, Buttons, Sliders
  - Position input
Outline: Part 4

• Three-Dimensional Graphics
• Text: Chapters 4-6
• Lecture 5-8
  • Geometry
  • Transformations
  • Homogeneous Coordinates
  • Viewing
  • Lighting and Shading
Outline: Part 5

• Discrete Methods
• Text: Chapter 7
• Lectures 9-11
  • Buffers
  • Pixel Maps
  • Texture Mapping
  • Compositing and Transparency
  • Off-Screen Rendering
Outline: Part 6

• Hierarchy and Procedural Methods
• Text: Chapters 9-10
• Lectures 11-12
• Tree Structured Models
  • Traversal Methods
  • Scene Graphs
  • Particle Systems
  • Agent Based Models
Outline: Part 7

• Advanced Rendering
• Text: Chapter 8
• Leture 15
1.3

• Example: Draw a triangle
  • Each application consists of (at least) two files
  • HTML file and a JavaScript file

• HTML
  • describes page
  • includes utilities
  • includes shaders

• JavaScript
  • contains the graphics
Coding in WebGL

• Can run WebGL on any recent browser
  • Chrome
  • Firefox
  • Safari
  • IE

• Code written in JavaScript

• JS runs within browser
  • Use local resources
Example: triangle.html
Example Code

```html
<!DOCTYPE html>
<html>
<head>
<script id="vertex-shader" type="x-shader/x-vertex">
attribute vec4 vPosition;
void main(){
  gl_Position = vPosition;
}
</script>
<script id="fragment-shader" type="x-shader/x-fragment">
precision mediump float;
void main(){
  gl_FragColor = vec4( 1.0, 0.0, 0.0, 1.0 );
}
</script>
</head>
</html>
```
<script type="text/javascript" src="../Common/webgl-utils.js"></script>
<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="../Common/MV.js"></script>
<script type="text/javascript" src="triangle.js"></script>
</head>
<body>
<canvas id="gl-canvas" width="512" height="512">
Oops ... your browser doesn't support the HTML5 canvas element
</canvas>
</body>
</html>
var gl;
var points;

window.onload = function init()
{
    var canvas = document.getElementById( "gl-canvas" );
    gl = WebGLUtils.setupWebGL( canvas );
    if ( !gl ) { alert( "WebGL isn't available" );
}

    // Three Vertices

    var vertices = [
        vec2( -1, -1 ),
        vec2(  0,  1 ),
        vec2(  1, -1 )
    ];
// Configure WebGL
//
gl.viewport( 0, 0, canvas.width, canvas.height );
gl clearColor( 1.0, 1.0, 1.0, 1.0 );

// Load shaders and initialize attribute buffers

var program = initShaders( gl, "vertex-shader", "fragment-shader" );
gl.useProgram( program );

// Load the data into the GPU

var bufferId = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, bufferId );
gl.bufferData( gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW );
// Associate out shader variables with our data buffer

var vPosition = gl.getAttribLocation( program, "vPosition" );
gl.vertexAttribPointer( vPosition, 2, gl.FLOAT, false, 0, 0 );
gl.enableVertexAttribArray( vPosition );
render();

function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );
    gl.drawArrays( gl.TRIANGLES, 0, 3 );
}
Exercise

• Download the Code Examples from the course website

• Load the triangle.html and triangle.js to your computer and run them from there

• Edit the two files to change the color and display more than one triangle
JavaScript Notes

• JavaScript (JS) is the language of the Web
  • All browsers will execute JS code
  • JavaScript is an interpreted object-oriented language

• References
  • Flanagan, JavaScript: The Definitive Guide, O’Reilly
  • Crockford, JavaScript, The Good Parts, O’Reilly
  • Many Web tutorials
JS Notes

• Is JS slow?
  • JS engines in browsers are getting much faster
  • Not a key issues for graphics since once we get the data to the GPU it doesn’t matter how we got the data there

• JS is a (too) big language
  • We don’t need to use it all
  • Choose parts we want to use
  • Don’t try to make your code look like C or Java
JS Notes

• Very few native types:
  • numbers
  • strings
  • booleans

• Only one numerical type: 32 bit float
  • var x = 1;
  • var x = 1.0; // same
  • potential issue in loops
  • two operators for equality == and ===

• Dynamic typing
JS Arrays

- JS arrays are objects
  - inherit methods
  - var a = [1, 2, 3];
    is not the same as in C++ or Java
  - a.length // 3
  - a.push(4); // length now 4
  - a.pop(); // 4
  - avoids use of many loops and indexing
  - Problem for WebGL which expects C-style arrays
Typed Arrays

JS has typed arrays that are like C arrays

```javascript
var a = new Float32Array(3)
var b = new Uint8Array(3)
```

Generally, we prefer to work with standard JS arrays and convert to typed arrays only when we need to send data to the GPU with the flatten function in MV.js
A Minimalist Approach

• We will use only core JS and HTML
  • no extras or variants
• No additional packages
  • CSS
  • JQuery
• Focus on graphics
  • examples may lack beauty
• You are welcome to use other variants as long as I can run them from your URL
What is Computer Graphics?
Computer Graphics

- *Computer graphics* deals with all aspects of creating images with a computer
  - Hardware
  - Software
  - Applications
Example

• Where did this image come from?

• What hardware/software did we use to produce it?
Preliminary Answer

• **Application**: The object is an artist’s rendition of the sun for an animation to be shown in a domed environment (planetarium)

• **Software**: Maya for modeling and rendering but Maya is built on top of OpenGL

• **Hardware**: PC with graphics card for modeling and rendering
Basic Graphics System

Image formed in frame buffer

Input devices

Output device

• Computer graphics goes back to the earliest days of computing
  • Strip charts
  • Pen plotters
  • Simple displays using A/D converters to go from computer to calligraphic CRT

• Cost of refresh for CRT too high
  • Computers slow, expensive, unreliable
Cathode Ray Tube (CRT)

Can be used either as a line-drawing device (calligraphic) or to display contents of frame buffer (raster mode)
Shadow Mask CRT

- **Wireframe** graphics
  - Draw only lines
- Sketchpad
- Display Processors
- Storage tube

wireframe representation of sun object
Sketchpad

• Ivan Sutherland’s PhD thesis at MIT
  • Recognized the potential of man-machine interaction
  • Loop
    • Display something
    • User moves light pen
    • Computer generates new display
  • Sutherland also created many of the now common algorithms for computer graphics
Display Processor

- Rather than have the host computer try to refresh display use a special purpose computer called a display processor (DPU)

- Graphics stored in display list (display file) on display processor
- Host compiles display list and sends to DPU

• Raster Graphics
• Beginning of graphics standards
  • IFIPS
    • GKS: European effort
      • Becomes ISO 2D standard
    • Core: North American effort
      • 3D but fails to become ISO standard
• Workstations and PCs
Raster Graphics

• Image produced as an array (the *raster*) of picture elements (*pixels*) in the *frame buffer*
Raster Graphics

• Allows us to go from lines and wire frame images to filled polygons
PCs and Workstations

• Although we no longer make the distinction between workstations and PCs, historically they evolved from different roots
  • Early workstations characterized by
    • Networked connection: client-server model
    • High-level of interactivity
  • Early PCs included frame buffer as part of user memory
    • Easy to change contents and create images

Realism comes to computer graphics

smooth shading  environment mapping  bump mapping

- Special purpose hardware
  - Silicon Graphics geometry engine
    - VLSI implementation of graphics pipeline
- Industry-based standards
  - PHIGS
  - RenderMan
- Networked graphics: X Window System
- Human-Computer Interface (HCI)

• OpenGL API

• Completely computer-generated feature-length movies (Toy Story) are successful

• New hardware capabilities
  • Texture mapping
  • Blending
  • Accumulation, stencil buffers

- Photorealism
- Graphics cards for PCs dominate market
  - Nvidia, ATI
- Game boxes and game players determine direction of market
- Computer graphics routine in movie industry: Maya, Lightwave
- Programmable pipelines
- New display technologies
Generic Flat Panel Display

Vertical grid

Light emitting elements

Horizontal grid
Computer Graphics 2011–

• Graphics is now ubiquitous
  • Cell phones
  • Embedded

• OpenGL ES and WebGL

• Alternate and Enhanced Reality

• 3D Movies and TV
Image Formation
Objectives

• Fundamental imaging notions
• Physical basis for image formation
  • Light
  • Color
  • Perception
• Synthetic camera model
• Other models
Image Formation

• In computer graphics, we form images which are generally two dimensional using a process analogous to how images are formed by physical imaging systems
  • Cameras
  • Microscopes
  • Telescopes
  • Human visual system
Elements of Image Formation

• Objects
• Viewer
• Light source(s)

• Attributes that govern how light interacts with the materials in the scene
• Note the independence of the objects, the viewer, and the light source(s)
Light

• *Light* is the part of the electromagnetic spectrum that causes a reaction in our visual systems

• Generally these are wavelengths in the range of about 350-750 nm (nanometers)

• Long wavelengths appear as reds and short wavelengths as blues
Ray Tracing and Geometric Optics

One way to form an image is to follow rays of light from a point source finding which rays enter the lens of the camera. However, each ray of light may have multiple interactions with objects before being absorbed or going to infinity.
Luminance and Color Images

• Luminance Image
  • Monochromatic
  • Values are gray levels
  • Analogous to working with black and white film or television

• Color Image
  • Has perceptual attributes of hue, saturation, and lightness
  • Do we have to match every frequency in visible spectrum? No!
Three-Color Theory

• Human visual system has two types of sensors
  • Rods: monochromatic, night vision
  • Cones
    • Color sensitive
    • Three types of cones
    • Only three values (the *tristimulus* values) are sent to the brain

• Need only match these three values
  • Need only three *primary* colors
Shadow Mask CRT
Additive and Subtractive Color

• Additive color
  • Form a color by adding amounts of three primaries
    • CRTs, projection systems, positive film
    • Primaries are Red (R), Green (G), Blue (B)

• Subtractive color
  • Form a color by filtering white light with cyan (C), Magenta (M), and Yellow (Y) filters
    • Light-material interactions
    • Printing
    • Negative film
Pinhole Camera

Use trigonometry to find projection of point at \((x,y,z)\)

\[
\begin{align*}
    x_p &= -x/z/d \\
    y_p &= -y/z/d \\
    z_p &= d
\end{align*}
\]

These are equations of simple perspective
Synthetic Camera Model

projector

image plane

projection of p

center of projection

Angel and Shreiner: Interactive Computer Graphics 7E
© Addison-Wesley 2015
Advantages

- Separation of objects, viewer, light sources
- Two-dimensional graphics is a special case of three-dimensional graphics
- Leads to simple software API
  - Specify objects, lights, camera, attributes
  - Let implementation determine image
- Leads to fast hardware implementation
Global vs Local Lighting

• Cannot compute color or shade of each object independently
  • Some objects are blocked from light
  • Light can reflect from object to object
  • Some objects might be translucent
Why not ray tracing?

• Ray tracing seems more physically based so why don’t we use it to design a graphics system?
• Possible and is actually simple for simple objects such as polygons and quadrics with simple point sources
• In principle, can produce global lighting effects such as shadows and multiple reflections but ray tracing is slow and not well-suited for interactive applications
• Ray tracing with GPUs is close to real time