Books, OpenGL, GLUT, CUDA, OpenCL, OpenCV, PointClouds, G3D, and Qt

CS334 Fall 2017

Daniel G. Aliaga
Department of Computer Science
Purdue University
Books
(and by no means complete...)

- “Interactive Computer Graphics”
  - Angel and Shreiner, pub: Addison Wesley

- “3D Computer Graphics”
  - Watt, pub: Addison Wesley

- “Real-time Rendering”
  - Moller and Haines, pub: AK Peters

- “3D Game Engine Design”
  - Eberly, pub: Morgan Kaufmann

- “Level of Detail for 3D Graphics”
  - Luebke, Reddy, Cohen, Varshney, Watson, and Huebner, pub: Morgan Kaufmann

  - Hughes, van Dam, McGuire, Sklar, Foley, Feiner, and Akeley
## Computer Graphics Pipeline

<table>
<thead>
<tr>
<th>Geometric Primitives</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Transformation</td>
<td>Transform into 3D world coordinate system</td>
</tr>
<tr>
<td>Lighting</td>
<td>Simulate illumination and reflectance</td>
</tr>
<tr>
<td>Viewing Transformation</td>
<td>Transform into 3D camera coordinate system</td>
</tr>
<tr>
<td>Clipping</td>
<td>Clip primitives outside camera’s view</td>
</tr>
<tr>
<td>Projection Transformation</td>
<td>Transform into 2D camera coordinate system</td>
</tr>
<tr>
<td>Scan Conversion</td>
<td>Draw pixels (incl. texturing, hidden surface…)</td>
</tr>
</tbody>
</table>

**Image**
Linear Algebra

• Why do we need it?
  – Modeling transformation
    • Move “objects” into place relative to a world origin
  – Viewing transformation
    • Move “objects” into place relative to camera
  – Perspective transformation
    • Project “objects” onto image plane
OpenGL

• Software interface to graphics hardware
• ~150 distinct commands
• Hardware-independent and widely supported
  – To achieve this, no windowing tasks are included
• GLU (Graphics Library Utilities)
  – Provides some higher-level modeling features such as curved surfaces, objects, etc.
• Open Inventor (old)
  – A higher-level object-oriented software package
OpenGL Online

• Current version is: ~4.5
• Website
  – http://www.opengl.org
• Books
  – Programming Guide ("Red book")
OpenGL

• Rendering parameters
  – Lighting, shading, lots of little details...

• Texture information
  – Texture data, mapping strategies

• Matrix transformations
  – Projection
  – Model view
  – (Texture)
  – (Color)
Matrix Transformations
Matrix Transformations

• Each of modelview and projection matrix is a 4x4 matrix
• OpenGL functions
  – glMatrixMode(…)
  – glLoadIdentity(…)
  – glLoadMatrixf(…)
  – glMultMatrix(…)
  – glTranslate(…)
  – glScale(…)
  – glRotate(…)
  – glPushMatrix()
  – glPopMatrix()
Matrix Transformations

{

...  
...  
...  

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(N); /* apply transformation */
glMultMatrixf(M); /* apply transformation M */
glMultMatrixf(L); /* apply transformation L */
glBegin(GL_POINTS);
  glVertex3f(v); /* draw transformed vertex v */
glEnd();
...  
...  
...  
}

= draw transformed point “N(M(Lv))”
Modelview Transformations

\texttt{glRotatef(45,0,0,1)}

\texttt{glTranslate3f(tx,ty,tz)}

\texttt{glScalef(2,-0.5,1.0)}
Modelview Transformations

glRotatef(d, rx, ry, rz);
glTranslate3f(tx, ty, tz);

glTranslate3f(tx, ty, tz);
glRotatef(d, rx, ry, rz);
Simple OpenGL Program

{
  <Initialize OpenGL state>

  <Load and define textures>

  <Specify lights and shading parameters>

  <Load projection matrix>

  For each frame

    <Load model view matrix>
    <Draw primitives>

  End frame

}
Simple OpenGL Program

```c
#include <GL/gl.h>
main()
{
    InitializeAWindowPlease();
    glMatrixMode(GL_PROJECTION);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslate3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();
    UpdateTheWindowAndCheckForEvents();
}"
GLUT/FreeGLUT

• = Graphics Library Utility Toolkit
  – Adds functionality such as windowing operations to OpenGL

• Event-based callback interface
  – Display callback
  – Resize callback
  – Idle callback
  – Keyboard callback
  – Mouse movement callback
  – Mouse button callback
Simple OpenGL + GLUT Program

#include <…>

DisplayCallback()
{
  <Clear window>
  <Load Projection matrix>
  <Load Modelview matrix>
  <Draw primitives>
  (<Swap buffers>)
}

IdleCallback()
{
  <Do some computations>
  <Maybe force a window refresh>
}

KeyCallback()
{
  <Handle key presses>
}

MouseMovementCallback
{
  <Handle mouse movement>
}

MouseButtonsCallback
{
  <Handle mouse buttons>
}

Main()
{
  <Initialize GLUT and callbacks>
  <Create a window>
  <Initialize OpenGL state>

  <Enter main event loop>
}
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}

void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glLoadIdentity ();
    gluLookAt (0, 0, 5, 0, 0, 0, 0, 1, 0);
    glScalef (1.0, 2.0, 1.0);
    glutWireCube (1.0);
    glFlush ();
}

int main(int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
Simple OpenGL + GLUT Program
Example Program with Lighting

```c
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

void init(void)
{
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 50.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_SMOOTH);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}

void display(void)
{
    glEnable(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glutSolidSphere (1.0, 20, 16);
    glutFlush ();
}

void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
                1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
    else
        glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
                 1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```
CUDA and OpenCL

• NVIDIA defined “CUDA” (new)
  – Compute Unified Device Architecture
  – http://www.nvidia.com/object/cuda_home.html#

• Khrono’s group defined “OpenCL” (newer)
  – Open Standard for Parallel Programming of Heterogeneous Systems
  – http://www.khronos.org/opencl/
CUDA Example

- Rotate a 2D image by an angle
  - On the CPU (PC)
    - simple-tex.pdf
  - On the GPU (graphics card)
    - simple-tex-kernel.pdf
OpenCL Example

• Compute a Fast Fourier Transform
  – On the CPU (PC)
    • cl-cpu.pdf
  – On the GPU (graphics card)
    • cl-gpu.pdf
OpenCV

• A library for computer-vision related software
• Derived from research work and high-performance code from Intel
• http://opencv.willowgarage.com/wiki/
Point Clouds

• For 3D point cloud processing and rendering
  – http://www.pointclouds.org
Microsoft XNA

• A Visual Studio programming environment to create games for Windows Phone, Xbox 360, and Windows-based computers
Unity 3D

- Unity is a game development ecosystem: a powerful rendering engine fully integrated with a complete set of intuitive tools and rapid workflows to create interactive 3D content; easy multiplatform publishing; thousands of quality, ready-made assets in the Asset Store and a knowledge-sharing Community.

The G3D Innovation Engine is

- an open-source commercial-grade C++ 3D engine
- used in commercial games, research papers, military simulators, and university courses.
- supports hardware accelerated real-time rendering, off-line rendering like ray tracing, and general purpose computation on GPUs.

[http://g3d.sourceforge.net](http://g3d.sourceforge.net)
Qt

- Qt is a cross-platform application framework, including OpenGL support, that can be run on various software and hardware platforms

www.qt.io