Course Description

Modeling, viewing, and rendering techniques in 3D computer graphics systems. Topics include:

• modeling systems and data structures;
• polygonal and parametric surface representation;
• 3D transformations and introduction to 3D animation;
• the synthetic camera paradigm;
• 3D clipping and projections;
• hidden surface removal algorithms;
• techniques for realism such as textures, materials, lighting, shadows, terrain, and atmospheric effects;
• normal, bump, and height mapping;
• tessellation; procedural models and fractals.

Emphasis on hardware support and shader pipeline programming.
Parallel and Perspective Projections:

Hidden Surface Removal, Clipping, Culling:

Color, Lighting, Materials:

Shadows:

Textures:

Reflection, Refraction, Transparency:
Terrain, Tessellation:

Height mapping, Bump mapping, Normal mapping:

Environment Mapping:

Fractals:

This is not an easy class:

- “Bleeding edge” technology (OpenGL4, JOGL)
  - Technology changes quickly and often
  - Most instructional resources are outdated
  - Books, websites, are rushed, full of typos/errors
  - Hardware, drivers, always full of bugs
  - Machines & graphics cards differ in details
- Debugging shader code is very hard
  - Not many debugging tools for shader code
  - No way to “print” variables
  - Instructor may not have time to help everyone
- Expect to put in a lot of time
- Expect to have to solve many problems yourself!
Prerequisites

• CSc 133 (Obj-Orient. Comp. Graphics Prog.)
  … which means you must also have:
  ▪ CSc 15 (Programming Methodology I)
  ▪ CSc 20 (Programming Methodology II)
  ▪ CSc 28 (Discrete Structures)
  ▪ CSc 130 (Algorithms and Data Structures)
  ▪ CSc 131 (Introduction to Software Engineering)
  ▪ Math 29 (Pre-calculus Mathematics)

• Also very helpful:
  ▪ Math 30 or 26A (Calculus I)
  ▪ Physics 11A or 5A (Mechanics)

Required Text:
• OpenGL SuperBible, 6th or 7th Ed., Graham Sellers, et. al., Addison-Wesley
• JOGL Workbook, Gordon and Clevenger, draft (provided free)

Useful References:
• Computer Graphics: Theory Into Practice, Jeffrey McConnell, Jones & Bartlett Publishers

Prerequisites By Topic

Programming Experience:
- Algorithms and Data Structures
  ▪ Linear Lists, Stacks, Queues, Binary Trees, Recursion
  ▪ OOP principles
  ▪ Abstraction, Encapsulation, Inheritance, Polymorphism
  ▪ Design Patterns
  ▪ Java Interface Types
  ▪ Basic GUI Creation
    ▪ Menus, Buttons, Frames, Panels, other “widgets”
  ▪ Event-Driven Programming
  ▪ Interactive Operations
    ▪ Mouse, Keyboard, and Timer Event Handling
    ▪ Screen-based animation and repaint() operations

Math:
- Matrix Concepts and Operations
  ▪ Matrix addition & multiplication by scalars and vectors
  ▪ Row-major vs. column-major representation; multiplication from the left” vs. “from the right”
  ▪ Multiplication by another matrix: associativity (yes), commutativity (no)
  ▪ Transpose & Inverse
- Vector Concepts and Operations
  ▪ Definition of a vector; as a single point, or diff. of two points
  ▪ Equivalency after translation
  ▪ Computing Magnitude (length)
  ▪ Multiplication by a scalar value; effect of multiplying by a fractional and/or negative value
  ▪ Definition and computation of Unit vectors
  ▪ Definition and Computation of the Dot Product

Graphical Operations:
- Homogeneous representation of 2D point (x,y) as [ x  y  1 ]
- Matrix Representation for (2D) Translation, Scaling, and Rotation
  ▪ Row vs. Column Forms
  ▪ Proper order of operations
- Application of Transformation Matrices to Homogeneous Points
- Concatenation of Transformations
  ▪ Combining multiple transformations into one
  ▪ Use of translation plus rotation about the origin to accomplish arbitrary rotation
- Coordinate Systems
  ▪ Local, world, device
- World windows vs. viewports, VTM
- See CSc 133 Lecture Notes – “Transformations” & “Viewing”

Types & Algorithms:
- Color Spaces
  ▪ RGB “Color Cube” & Java Color type
- Java2D AffineTransform type
- Points, Lines, and Line Rasterization
- Cubic Bezier Curves
  ▪ Parametric Representation of lines
  ▪ Control points
  ▪ Blending functions
  ▪ Geometric representation
  ▪ Analytic derivation
  ▪ Recursive drawing algorithm
- See CSc 133 Lecture Notes
  ▪ “Applications of Affine Transforms”, “Lines & Curves”
Cubic Bezier Curves – Review

Given 4 “control points” [P0, P1, P2, P3]:
• Draw the “control mesh” (connect the control points)
• Connect all points of equal “parametric value”
• Points with parametric value ‘t’ are on the curve

Control Mesh Subdivision

• Splitting a control mesh [Q] at t=1/2 produces two meshes [R] and [S]

Subdivision Algorithm

//** Splits the input control point vector Q into two control point vectors R and S such that
// R and S define two Bezier curve segments that together exactly match the Bezier curve defined by Q.
void subdivideCurve (ControlPointVector Q, R, S) {
  /** Splits the input control point vector Q into two control point vectors R and S such that
   * R and S define two Bezier curve segments that together exactly match the Bezier curve
   * defined by Q.
   */

  // Check for degenerate case
  if (straightEnough (ControlPointVector)  OR  (Level > MAXLEVEL) ) {
    return true ;
  }

  // Otherwise, recursively subdivide
  subdivideCurve (ControlPointVector, LeftSubVector, RightSubVector) ;

  // Draw line from 1st control point to last control point
  draw line from 1st control point to last control point ;

  // Draw the “control mesh” (connect the control points)
  drawBezierCurve (LeftSubVector,Level+1) ;
  drawBezierCurve (RightSubVector,Level+1) ;
}

Analytic Derivation

Let P(t) be a point on the line P0–P3, and let P(t) be a point on the line P0–P3 (see previous diagram). Then

\[ P_{01}(t) = t \cdot P_0 + (1-t) \cdot P_1 \]  \[ P_{12}(t) = t \cdot P_1 + (1-t) \cdot P_2 \]  \[ P_{23}(t) = t \cdot P_2 + (1-t) \cdot P_3 \]

Likewise, let P(t) be a point on the line P2–P3(l), and P(t) is a point on the line P0–P1(l). Then,

\[ P_{12}(t) = t \cdot P_{12} + (1-t) \cdot P_{23} \]

Following this same procedure, substituting, and then solving for P(t) gives

\[ P(t) = (1-t)^3 \cdot P_0 + 3(1-t)^2 t \cdot P_1 + 3(1-t) t^2 \cdot P_2 + t^3 \cdot P_3 \]

Thus we say that the curve defined by [P0, P1, P2, P3] is the set of all points P(t) such that

\[ P(t) = \sum_i (1-t)^3 \cdot B_i(t) \cdot P_i \]

This Week

• Make sure you have Java 1.8

• If you are using your own machine:
  o make sure it supports OpenGL 4.3 or higher (use verification tool)
  o install JOGL (handout on course website)

• Install graphicslib3D (download from course website)

• Do the assigned readings on the course schedule.