Lecture 6

More Projections and Clipping

Introduction to OpenGL (Graphics Library)

Friday, February 4, 2000

William H. Hsu
Department of Computing and Information Sciences, KSU
http://www.cis.ksu.edu/~whsu

Readings:
Sections 3.12, 6.5-6.6, Foley et al/
Section 6.7, Hearn and Baker 2e
Chapter 2, Sections 4.9, 7.5-6, 7.7-7.6, Angel 2e

Clipping Lines

• Clipping Lines (Introduction)
  – Cohen-Sutherland algorithm
  – Cyrus-Beck / Liang-Baracky algorithm
  – Clipping in 3D
    – Extending 2D line clipping algorithms to 3D objects
      – Sketch (more later): clipping in homogeneous coordinates
    – Graphics libraries: history and design rationale
    – Specification of graphics libraries: application programmer interfaces (API)
      – Key OpenGL functions

3D Projections and Clipping

• Projections (Concluded)
  – Parallel projection: cuboid view volume
  – Perspective projection: truncated pyramidal view volume (frustum)
  – Problem: how to clip?

• Clipping
  – Given: coordinates for primitives (line segments, polygons, circles, ellipses, etc.)
  – Determine: visible components of primitives (e.g., line segments)
  – Methods
    – Solving simultaneous equations (quick rejection: testing endpoints)
    – Solving parametric equations
    – Objectives: efficiency (e.g., fewer floating point operations)

• Clipping in 3D
  – Some 2D algorithms extendible to 3D
  – Specification (and implementation) of view volumes needed

Normalizing Transformation for Parallel Projection

• \( N_{\parallel} \) Transformation (Corresponding to Stack of Primitive Matrix Ops)
  – 4-Step Transformation (Section 6.5.1, FVD)
    – [1] VRP → origin
      – Translate “at point” to origin
      – Purpose: normalization for impending rotation
    – [2] Rotate \((x, y, z)\) to \((u, v, n)\)
      – Align VRC with WC
      – Purpose: normalize directional frame of reference according to viewer
      – Apply \( S_{\parallel} \)
      – Purpose: align center line of view volume with z axis (Figure 6.49, FVD)
    – [4] Translate and scale to canonical parallel cuboid
      – Nonuniform scaling according to \( u/v \) range (Equation 6.36, FVD)
      – Purpose: normalizes dimensions of view volume (Equation 6.36, FVD)
  – Result
    – \( N_{\parallel} = R_{\parallel} \cdot T_{\parallel} \cdot S_{\parallel} \cdot R_{\parallel}^{-1} \cdot T_{VRP} \)
    – Equation 6.36, FVD

Clipping Lines

• Clipping (Sections 3.11-3.12, 6.5.3-6.5.4, FVD; Sections 7.2-7.6, Angel)
  – Problem
    – Input: coordinates for primitives
    – Output: visible components of primitives
    – Equational solutions: simultaneous, parametric:
      – Basic primitive: clip individual points (test against rectangle bounds)
  – Lines (Section 3.12, FVD; Section 7.3, Angel)
    – Clipping line segment \( AB \) against rectangular plane \( R \)
      – General idea 1 (equational / regional approach)
        – Divide plane into regions about \( R \), see whether \( AB \) can possibly intersect
        – Find intersections
      – General idea 2 (parametric approach)
        – Express line as parametric equation(s): 1 matrix or 2 scalar
        – Find intersections by plugging into parametric equation (Table 3.1, FVD)
        – Use to check clipping cases

Normalizing Transformation for Perspective Projection

• \( N_{\perp} \) Transformation (Corresponding to Stack of Primitive Matrix Ops)
  – 5-Step Transformation (Section 6.5.2, FVD)
    – [1] VRP → origin
      – Translate “at point” to origin
      – Purpose: normalization for impending rotation
    – [2] Rotate \((x, y, z)\) to \((u, v, n)\)
      – Align VRC with WC
      – Purpose: normalize directional frame of reference according to viewer
      – Translate “eye” to origin
      – Purpose: normalize position of reference according to viewer
      – Apply \( S_{\perp} \)
      – Purpose: align center line of view volume with z axis (Figure 6.53, FVD)
    – [5] Scale to canonical perspective frustum
      – Nonuniform scaling according to ratio of sheared-z to \( u/v \) range (Equation 6.35, FVD)
      – Purpose: normalize dimensions of view volume (Equation 6.23, FVD)
Cohen-Sutherland Algorithm

- **General Idea 1** [Cohen and Sutherland, 1963]
  - Divide plane into 9 regions about and including R
  - See whether AB can possibly intersect
- **Outcodes:** Quick Rejection Method for Intersection Testing
  - Unique 4-bit binary number for each of 9 regions
    - \( b_0 = 1 \) iff \( y > y_{\text{max}} \)
    - \( b_1 = 1 \) iff \( y < y_{\text{min}} \)
    - \( b_2 = 1 \) iff \( x > x_{\text{max}} \)
    - \( b_3 = 1 \) iff \( x < x_{\text{min}} \)
- Check clipping cases
  - 8 floating-point subtractions per line segment, plus integer comparison
  - Each line segment has 2 outcodes: \( o_1, o_2 \)
- Case 1: \( o_1 = o_2 = 0000 \) – inside; show whole segment
- Case 2: \( o_1 = 0000, o_2 \neq 0000 \) (or vice versa) – partly inside; shorten
- Case 3: \( o_1 \neq 0000, o_2 \neq 0000 \) – totally outside; discard
- Case 4: \( o_1 \neq 0000, o_2 \neq 0000 \) – both endpoints outside; check further

Cyrus-Beck and Liang-Barsky Algorithms

- **General Idea 2** [Cyrus and Beck; Liang and Barsky]
  - Express line as parametric equation(s): 1 matrix or 2 scalar
  - Find intersections by plugging into parametric equation (Table 3.1, FVD)
  - Use to check clipping cases
- **Cyrus-Beck Algorithm**
  - Section 3.12.4, FVD
  - More details next class (Lecture 7)
- **Liang-Barsky Algorithm**
  - Section 3.12.4, FVD; Section 7.3.2, Angel
  - More details next class (Lecture 7)

View Volumes in 3D: Perspective Frustum and Parallel Cuboid

- View volume (frustum)
  - Based on Figure 7.21, [Angel, 2000]
- Based on Figure 5.21, and Figure 13-30(b) [Hearn and Baker, 1997]
- Center of Projection (COP)

Generic Graphics Package: Overview

- **Turn to Your Partners**
  - People in your row
  - Groups numbered counterclockwise (left front to right front)
- **Exercise 1 (Now): Generic Graphics Package**
  - Objective: understanding generic graphics kernels
  - Exercise (5 minutes): list
    - 3 logical groups of functions that simple graphics kernels have
    - 1 criterion for deciding whether kernel function should be implemented in hardware, software, or as macro
- **Exercise 2 (Later Today): Specifying Graphics Transformations**
  - Objective: understanding shear transformation
  - Specification of shear transformation function
  - Implementation in OpenGL
- **Exercise 3 (Later Today): Applying Graphics Transformations**
  - Objective: using shear to implement one type of parallel projection from another
  - Enhancing capabilities of OpenGL

In-Class Exercises (TTYP): Generic Graphics Package

- **Graphics Kernels**
  - GKS
  - PHIGS (FVD)
  - OpenGL
- **Generic Graphics Package**
  - Specification
    - Requirements analysis: deciding what to include
    - Design of object model
    - Implementation
      - In hardware
      - In software (part of kernel)
      - As macros (part of kernel)
      - By application programmer

Generic Graphics Package: Typical Components

- **TTYP Exercise 1a: Typical Components of Generic Graphics Kernels**
  - 1. Scan conversion
  - 2. Transformations
  - 3. Clipping
  - 4. View specification / rendering
  - 5. Texturing / mapping
  - 6. 2-D primitives
  - 7. Illumination
  - 8. Color
- **What Else?**
  - 1. Animation
  - 2. Event handling (GUI)
  - 3. Window management
• TTYP Exercise 1b: Criteria for Implementation
  – In hardware
  • 1. Frequently used
  • 2. Need fast implementation
  – Library macro
  • 1. Fast
  • 2. Small, but frequently used
  – In software (library function)
  • 1. Save space (memory intensive), but not as frequently used
  • 2. Portability (possibility platform / OS dependent)
  – By applications programmer(s)
  • 1. Infrequently used but important to end-user
  • 2. Nonstandard techniques or requirements
• How Else Can We Decide At What “Level” To Place Functions?
  – 1. Cost issues: speed / frequency of use (generality of purpose) tradeoffs
  – 2. Programming language: what are non-graphical primitives?

OpenGL: Transformation Matrices
• OpenGL Matrix Stack (Section 4.9, Angel)
  – General syntax: glMatrixOperation (parameters)
  – Loading
    • glLoadMatrixf (pointer-to-matrix)
    • Special case: glLoadIdentity ()
  – Implicit parameter: “currently loaded matrix”
    • e.g., glLoadIdentity (); glTranslatef (90.0, 1.0, 0.0);           /* 90 degrees roll */
    • NB: convention – postmultiplication
• Translation
  – Syntax: glTranslatef (dx, dy, dz)
• Rotation
  – Syntax: glRotatef (angle, vx, vy, vz)
  – vx, vy, vz: not gltf, yz components
• Scaling
  – Syntax: glScalef (sx, sy, sz)
• Shearing
  – TTYP Exercise... Write glShear (parameters)
  – glLoadIdentity ()
  – Special case: glLoadIdentity ()

OpenGL: Orthographic and Oblique Projections
• Orthographic Projections in OpenGL (Section 5.7, Angel)
  – Orthographic: only parallel projections provided by OpenGL
  – Procedure
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho (-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);   /* canonical view volume */
  – General syntax: glOrtho (xmin, xmax, ymin, ymax, zmin, zmax)
• Implementing Oblique Projections
  – Problem: OpenGL provides only pure orthographic projections
  – Case where VPN (and projectors) its principal face normal
  • Top, front, side elevations
  – Solution
    • Q: How to implement oblique projection using glOrtho?
      • A: Use shear transformation (Chapter 6, FVD; 5.7.2 Angel... Homework 2)
    • TTYP exercise: use your glShearf to do this

OpenGLOpenGL:: OpenGL
Department of Computing and Information Sciences
CIS 736: Computer Graphics
Kansas State University

OpenGLOpenGL:: OpenGL
Department of Computing and Information Sciences
CIS 736: Computer Graphics
Kansas State University

TTYP Exercise 1b: Criteria for Implementation
In hardware
• 1. Frequently used
• 2. Need fast implementation
Libraries
• 1. Fast
• 2. Small, but frequently used
In software (library function)
• 1. Save space (memory intensive), but not as frequently used
• 2. Portability (possibility platform / OS dependent)
By applications programmer(s)
• 1. Infrequently used but important to end-user
• 2. Nonstandard techniques or requirements

How Else Can We Decide At What “Level” To Place Functions?
1. Cost issues: speed / frequency of use (generality of purpose) tradeoffs
2. Programming language: what are non-graphical primitives?

OpenGL: Overview of Utility Toolkit (GLUT)
• OpenGL Library Utility Toolkit (GLUT)
  – Chapter 2, Angel
  – Supplements and related links: http://www.cs.umn.edu/~angel
  – Links to web resources, code examples: http://www.cs.umn.edu/~angel
  – Programs from book: ftp.cs.umn.edu (pub/angel/BOOK)
• Color
  – Chapter 13, FVD; Section 2.4, Angel
  – More next month
• Viewing
  – Chapters 3 and 6, FVD; Section 2.5, Angel
• Window System
  – Chapter 3, FVD; Section 2.6, Angel
  – More in second half of CIS 736

OpenGL: Viewing API and Look-At Function
Recall: Viewing Reference Coordinate (VRC) System Specification
• World coordinates (x, y, z)
• Viewing coordinates (u, v, n)
  • n = view plane normal
  • u = projection of VUP (view-up vector), orthogonal to n, in view plane
  • v = third basis vector (orthogonal to n, v; can compute using cross product)

Look-At Function (Section 5.2.3, Angel)
• Syntax: gluLookAt (eyex, eyey, eyez, atx, aty, atz, upx, upy, upz)
• Parameters
  • eyex, eyey, eyez: specification of eyepoint (as discussed last week)
  • atx, aty, atz: specification of viewpoint (view reference point aka VRP)
  • upx, upy, upz: specification of view up vector (VUP)
• Properties of Viewing API
  • VPN = e - a
  • Specifies synthetic camera (as discussed last week)
• Now, Ready to Project...
Course Project: Overview

- 3 Components
  - Project proposal (20%, 50 points)
  - Implementation (50%, 125 points)
  - Final report (50%, 125 points)
- Project Proposal (Due 02/14/2000)
  - 1-3 page description of project topic, plan
  - Guidelines: next suggested topic, tools to appear on CIS 736 course web page
  - See: implementation practicum links (Brown, Cornell, UNC, others) on 736 page
- Implementation
  - Students choice of programming language
  - Guidelines: next Wednesday (and on 736 page)
- Final Report
  - 4-6 page report on implementation, experimental results, interpretation
  - Peer-reviewed (does not determine grade)
  - Reviews graded (short report worth 60 points, reviews worth 15 points)

Proposal Guidelines

- Report Contents (1-3 Pages)
  - Scope: What kind of CG algorithms will you use?
  - Problem: What display problem are you addressing?
  - Methodology: How are you addressing the problem?
  - Scopes
    - What rendering, animation, and visualization tools (or codes) will you use?
    - What characteristics of the display tools are you trying to deal with / exploit?
  - Problem
    - Objective: What is your display objective?
    - Evaluation: How will you demonstrate (and measure) success?
  - Methodology
    - Implementation: What will you implement? (general statement, not specification)
    - Graphics data representation: How will you manipulate and represent CG data?
    - Infrastructure: What programming languages and platform(s) will you use?
  - Objective: What is your display objective?
  - Methodology: How are you addressing the problem?
  - Evaluation: How will you demonstrate (and measure) success?
- Problem
  - Objective: What is your display objective?
  - Methodology: How are you addressing the problem?
  - Evaluation: How will you demonstrate (and measure) success?

Terminology

- Normalizing Transformations
  - $N_{pp}$: normalizing transformation for parallel projection (6.5.1, FVD)
  - $N_{conf}$: normalizing transformation for perspective projection (6.5.2, FVD)
  - $M$: conversion matrix from perspective to parallel view volume (6.5.4, FVD)
  - $N_{conf} = M \cdot S = SH_{PP} \cdot TL_{PP} \cdot R \cdot TL_{FRP}$ (Equation 6.49, FVD)
- Clipping: Determining Parts of Primitives to Display
  - Cohen-Sutherland: line clipping algorithm
    - Division of plane into 9 regions with (4-bit) outcodes
    - Testing endpoints of line segment
    - Parametric clipping: line / rectangle intersection using parametric equation
  - Cyrus-Beck: generalized convex 3D polyhedron
  - Liang-Barney: more efficient, specialized variant (upright 2D, 3D clip regions)
  - Clipping in 3D
    - OBB: truncated viewing pyramid used to clip after $N_{conf}$
    - Frustum: truncated viewing pyramid
- OpenGL: Multiplatform, Standardized Graphics Library and API

Summary Points

- Projections: Review of $N_{conf}$
  - $[1]$ VRP -> origin
  - $[2]$ Rotate (x, y, z) to (u, v, n)
  - $[3]$ COP -> origin
  - $[4]$ Shear view volume
  - $[5]$ Scale to canonical perspective frustum
- Clipping Lines: Cohen-Sutherland, Liang-Barsky (Cyrus-Beck)
- Clipping in 3D
  - Graphics libraries: history, design rationale, specification, APIs
  - Key OpenGL functions
- Course Projects: Overview
  - Next Lecture
    - More OpenGL (Sections 10.1-10.6, Angel)
      - Intro to cubic curves (11.1.3, 11.3.1-11.2.2, FVD; 10.6-10.8, Hearn and Baker)