Xi'an Jiaotong-Liverpool University
西交利物浦大學

CPT205 Computer Graphics

Revision

Lecture 13
2024-25

Yong Yue and Nan Xiang

Topics covered

Lecture No.	Topics
01	Introduction / Hardware and software
02	Mathematics for computer graphics
03	Graphics primitives
04	Geometric transformations
05	Viewing and projection
06	Parametric curves and surfaces
07	3D modelling
08	Hierarchical modelling
09	Lighting and materials
10	Texture mapping
11	Clipping
12	Hidden-surface removal

Graphics hardware and software

- Concept and applications of computer graphics

  all aspenes producing image e.g.

- Graphics Hardware
  - Input (touch, light, sound and vision), processing (CPU/GPU) and output devices (screen, data glovers, VR glasses, etc.), tablet, CRT, LCD)
  - Frame buffers
  - Pixels and screen resolution

- Graphics Software
  - Focus: Techniques (low-level, e.g. algorithms and procedures)
  - Programming library / API (OpenGL, JOGL, etc.)
  - Not our focus: High level systems (Maya, Studio Max, AutoCAD, etc.)

- About OpenGL – gl(), glu(), glut()

Int/CreatWindow/Display Func

ReshapFunc 回调 / MainLoop

gluPerspective 视角

LookAt 观察投影

Orth2D 正交

NurbsCurve 曲线

gl Begin/gl End ↳gl Vertex/Color/MatrixMode 矩阵/Enable

Mathematics for computer graphics

3D → maniplation → 2D.

transformation → projection

存 – 算 – 显示

- ➢ Computer representation of objects
- ➢ Cartesian co-ordinate system
- ➢ Geometry – Points, lines and angles 叉积(右手)
- ➢ Trigonometry
- ➢ Vectors, unit vectors and vector calculations
  - Magnitude and direction
  - Form of $\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$
  - Dot product ($\mathbf{V1}\bullet\mathbf{V2} = x1*x2 + y1*y2$ and $\mathbf{V1}\bullet\mathbf{V2} = |\mathbf{V1}||\mathbf{V2}|\cos(\alpha)$, so $\cos(\alpha) = ?$)
  - Cross product ($\mathbf{V1} \times \mathbf{V2} = |\mathbf{V1}||\mathbf{V2}|\sin(\alpha)\mathbf{n}$, thus $|\mathbf{V1} \times \mathbf{V2}| = |\mathbf{V1}||\mathbf{V2}|\sin(\alpha)$)
- ➢ Matrices and matrix calculations transpose matrix A^T
  - Dimensions (rows x columns)
  - Square, symmetric, identity and inverse matrices (if $\mathbf{A} \times \mathbf{B} = \mathbf{B} \times \mathbf{A} = \mathbf{I}$, then $\mathbf{A} = \mathbf{B}^{-1}$ and $\mathbf{B} = \mathbf{A}^{-1}$)
  - Multiplication: condition and resultant matrix ($\mathbf{M_1}(r_1, c_1) \times \mathbf{M_2}(r_2, c_2) = \mathbf{M_3}(r_1, c_2)$ where $c_1 = r_2$)

$r_1 \times c_1 = r_2 \times c_2$

Geometric primitives

- Primitives: points, lines and polygons $P(t) = (1-t)P_0 + tP_1$
- Algorithms
  - DDA (Digital Differential Analyser) for line generation $\Delta x = 1, \Delta y = \begin{cases} 0 \\ 1 \end{cases} \quad 0 \le |m| \le 1$
  - Bresenham line algorithm in brief midle point.
  - Use of symmetry to reduce computation for circle generation (computation of only one octant) $45^\circ, \frac{1}{r}, \begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$
- Polygons and triangles in brief
  - Polygon as an ordered set of vertices flat / smooth shading
  - Graphics hardware is optimised for processing points and flat polygons
  - Complex objects are decomposed into triangles (tessellation) because they are always flat
  - Polygon fill

antialiasing / Jaggies.

pologen fill

strip

Transformation pipeline and geometric transformations

- The transformation pipeline (Modelling -> Viewing -> Projection -> Normalisation -> Device) ✓
- Types of transformation
  - Translation
  - Rotation (about origin in 2D and an axis in 3D)
  - Scaling
  - Reflection (about an axis in 2D and a plane in 3D)
  - Shearing →

$$\begin{pmatrix}x'\\y'\\1\end{pmatrix} = \begin{pmatrix}1 & \cot\theta & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix}$$

- Homogeneous co-ordinate transformation matrices (4x4 in 3D)
  - For single transformations
  - Composite matrices (e.g. **T**·**R**·**S**)

(hx, hy, h)

glLoadM (m)

glMultMatrix(m)

- OpenGL functions

```
glMatrixMode(), glLoadIdentity(), glTranslate(), glRotate(),
glScale(), glPushMatrix(), glPopMatrix(),
```

CTM (current transformation matrix)

stack -

Viewing and projection

- Types of projection
  - Planar geometric projection
  - Parallel orthogonal
  - Perspective realistic, equldistance
  - Advantages and disadvantages limit of perspective, easy to construct
- Orthogonal projection
- Frustum / Perspective projection
- Parameters for 3D viewing and projection viewplane (normal vector -Zx)
  - Camera position, look-at point, view-up vector for x_{view}-axis
  - Viewing volume, near and far clipping planes
  - Viewing plane / clipping window
- Texture / Projection
- OpenGL functions 一样 angle in y eye, lookat, up

```
glMatrixMode(), glFrustum()/gluPerspective(), gluLookAt(),
glOrtho(), gluOrtho2D()
```

fov, w/h, near, far 3 3 3

Parametric curves and surfaces

- Why parametric (explicit) representation? t
- Parametric curves
  - In 2D: $x = x(t), y = y(t), \quad (0 \le t \le 1)$
  - In 3D: $x = x(t), y = y(t), z = z(t), \quad (0 \le t \le 1)$

2D Line: $x = x_1 + t(x_2 - x_1),$
$y = y_1 + t(y_2 - y_1), (0 \le t \le 1)$

2D Circle: $x = r\cos(360t),$
$y = r\sin(360t), \quad (0 \le t \le 1)$

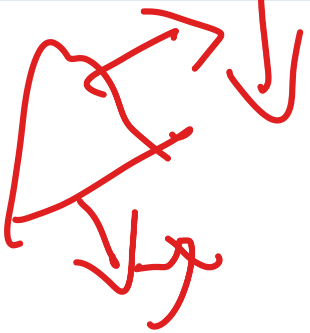
- Cubic curves and splines
  - Control points
  - Interpolation curves and design curves
  - Local control: tension and bias

n=k-1

$x(t) = a_0 + a_1t + a_2t^2 + a_3t^3, \quad (0 \le t \le 1)$
$y(t) = b_0 + b_1t + b_2t^2 + b_3t^3$

- Parametric surfaces
  - Revolved, extruded and swept surfaces
  - Tensor product surfaces
    - Interpolation surfaces and design surfaces
    - Controls

Splines



3D modelling

- Basic techniques: wireframes, surface models and solids; their advantages and disadvantages; their applications

- Constructive Solid Geometry (CSG)
  - CSG tree
  - Non-uniqueness

- Boundary Representation (B-Rep)
  - Boundary elements (geometry: points, curves and surfaces)
  - Relationships between boundary elements (topology / connectivity: vertices, edges and faces)
  - Types of B-Rep model:
    manifold (each vertex connecting at least 3 edges and each edge connecting exactly 2 faces) and non-manifold
  - Validity of B-Rep models and Euler's law ($V - E + F - R + 2H = 2S$)
  - Implementation of B-Rep models with OO techniques (e.g. C++)

Romulus/ACIS

Hierarchical modelling

➢ Important concepts

- Local and world co-ordinate frames of reference
- Object transformations

➢ Linear modelling

- Symbols (primitives): box, cone, cylinder, sphere, torus, etc.
- Instance and copy / array (linear and radial)
- Flat structure and no information of relationships between the parts

Aritculated

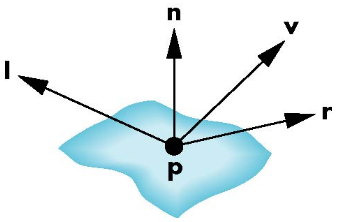
➢ Hierarchical modelling

- Hierarchical trees and articulated models
- Child inherits transformations from its parent ($\mathbf{M}_{\text{world}} = \mathbf{M}_{\text{parent}} \cdot \mathbf{M}_{\text{local}}$)
- While traversing the tree, **`glPushMatrix()`** is called when going down a level, and **`glPopMatrix()`** when going up
- Examples: car, robot, humanoid figure, solar system and train track

Lighting and materials

- Lighting sources and properties
  - Infinite distant and positional
  - Position, colour, direction, shape and reflectivity
- Lighting and material effects: ambient, diffuse and specular
- Attenuation for positional light sources
  - Idea situation: inversely proportional to the square of distance
  - Flexible implementation to include constant, linear and quadratic terms: $f(d) = \frac{1}{k_c + k_l d + k_q d^2}$
- Lighting model and shading
  - Phong model: 3 components, 4 vectors, 9 light intensity and 9 material absorption factors

$$I = k_d I_d \; \mathbf{l} \cdot \mathbf{n} \; + k_s I_s (\mathbf{v} \cdot \mathbf{r})^\alpha + k_a I_a$$



  - Polygon shading: constant (flat) and interpolative (smooth), Gouraud shading (averaging normal)
  - Combined effects: multiple lights, and light and materials
- OpenGL functions: `glLight{if}[v]()`, `glMaterial{if}[v]()`

value

Texture mapping

- Concepts and types of texture mapping
  - Texture mapping, environment mapping and bump mapping
  - Takes place at the end of the rendering pipeline
- Types of texture
  - 1D, 2D and 3D
  - Defined by an image file or a procedure
- 3 or 4 co-ordinate systems involved for texture mapping
- Secondary mapping (why and how)
- Control
  - Modes: decal, modulating and blending
  - Filtering: magnification and minification, and nearest or linear interpolation
  - Wrapping: repeating and clamping
- Multiple levels of detail – mipmapping (why and how)
- OpenGL functions

```
glTexImage2D(target,level,components,w,h,border,format,type,texels)
glTexParameteri(); glTexEnv{fi}[v](); glTexGen{ifd}[v]()
```

Clipping

- Concepts
  - Clipping window vs viewport (and display window)
  - Clipping primitives: points, lines, polygons, curves and text
- Line clipping algorithms
  - Brute Force Simultaneous Equations: slope-intercept formula does not handle vertical lines
  - Brute Force Similar Triangles: use of similar triangles to work out co-ordinates of intersect points with clipping window edges – computationally expensive!
  - Cohen-Sutherland
    - 9 regions of clipping plane, each with a 4-bit outcode
    - Logic OR and AND of outcodes for the two end points of the line
      - Trivial accept and trivial reject of a line
      - Calculate intersections based on bit values of the outcode when necessary
  - Liang-Barsky (parametric lines): value range of parameters for the 4 intersections determine if lines are to be accepted or rejected
- Polygon clipping in brief

Hidden-surface removal

- Concepts: clipping vs hidden-surface removal
- Object-space and image-space approaches
- Algorithms
  - Back-face culling
    - If $\mathbf{v}\bullet\mathbf{n} \geq 0$ (viewing direction and face normal), polygon is visible
    - Limitations: applicable to single and convex object for front-facing polygons
  - Painter’s algorithm: depth sort, overlap test in x and y directions, and hard cases
  - Binary Spatial Partition (BSP)
    - Structure and creation of BSP tree
    - Rendering of BSP tree
    - Non-unique
  - Z-buffer
    - A depth buffer is used in addition to frame buffer
    - For each pixel position, the polygon closest to the viewport is determined, and corresponding colour used
- OpenGL functions: `glutInitDisplayMode()`, `glClear()`, `glClearDepth()`, `glDepthRange()`; `glDepthFunc()`, `glDepthMask()`, `glPolygonMode()`, `glEnable(GL_DEPTH_TEST)`, `glEnable(GL_FOG)`, `glCullFace()`

About final exam

- Format of exam
  - 5 Questions, each worth 20 marks
  - Question 1 has 10 basic, short-answer questions
  - Questions 2-5 cover main topics of the module, and each has several (around 3) sub-questions
- Study a past exam paper
- Answer questions