University of British Columbia

CPSC 414 Computer Graphics

Projections and Picking

Wed 24 Sep 2003

- project 1 solution demo
- recap: projections 2
- projections 3
- picking

News

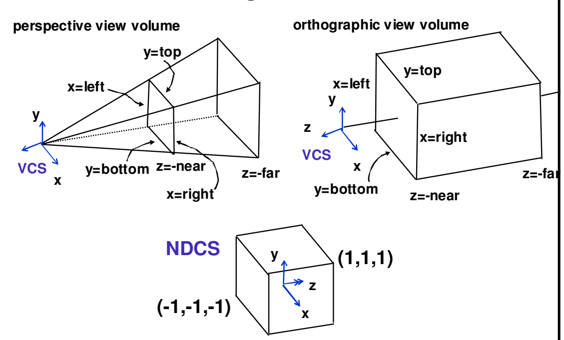
- Project 1 solution executable available
  - idea of what's expected
  - no need to copy look and feel exactly
- Readings reminder
  - Chapter 5

University of British Columbia

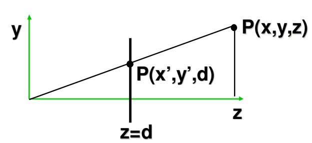
CPSC 414 Computer Graphics

Projections recap

Transforming View Volumes



Basic Perspective Projection



similar triangles

$$\frac{y'}{d} = \frac{y}{z} \rightarrow y' = \frac{y \cdot d}{z} \quad \textbf{also} \quad x' = \frac{x \cdot d}{z} \quad \textbf{but} \quad z' = d$$

- nonuniform foreshortening
  - not affine

Basic Perspective Projection

- can express as homogenous 4x4 matrix!

$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} \xrightarrow{/w} \begin{bmatrix} x \cdot d/z \\ y \cdot d/z \\ d \end{bmatrix}$$

Projective Transformations

- determining the matrix representation
  - need to observe 5 points in general position, e.g.
    - $[left,0,0,1]^T \rightarrow [-1,0,0,1]^T$
    - $[0,top,0,1]^T \rightarrow [0,1,0,1]^T$
    - $[0,0,-f,1]^T \rightarrow [0,0,1,1]^T$
    - $[0,0,-n,1]^T \rightarrow [0,0,-1,1]^T$
    - $[left*f/n, top*f/n, -f, 1]^T \rightarrow [-1,1,1,1]^T$
  - solve resulting equation system to obtain matrix

OpenGL Orthographic Matrix

- scale, translate, reflect for new coord sys

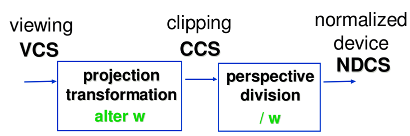
$$\begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

OpenGL Perspective Matrix

- shear, scale, reflect for new coord sys

$$\begin{bmatrix} \frac{2 \cdot near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2 \cdot near}{top-bot} & \frac{top+bot}{top-bot} & 0 \\ 0 & 0 & \frac{-(far+near)}{far-near} & \frac{-2 \cdot far \cdot near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Projection Normalization



- distort such that orthographic projection of distorted objects is desired persp projection
  - separate division from standard matrix multiplies
  - division: normalization

University of British Columbia

CPSC 414 Computer Graphics

Projections 3

Demos

- Brown Projections Demos
  - http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/viewing_techniques.html
- Nate Robbins tutorial (take 2):
  - http://www.xmission.com/~nate/tutors.html

Perspective Example

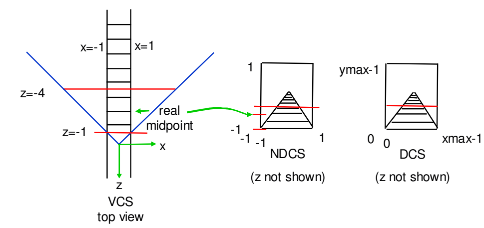
view volume
- left = -1, right = 1
- bot = -1, top = 1
- near = 1, far = 4

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -5/3 & -8/3 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspective Example

tracks in VCS:
left x=-1, y=-1
right x=1, y=-1

view volume
left = -1, right = 1
bot = -1, top = 1
near = 1, far = 4



Perspective Example

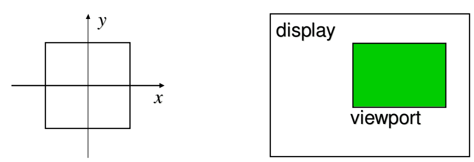
$$\begin{bmatrix} 1 \\ -1 \\ -5z_{VCS}/3-8/3 \\ -z_{VCS} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -5/3 & -8/3 \\ & & -1 & \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ z_{VCS} \\ 1 \end{bmatrix}$$

/ w

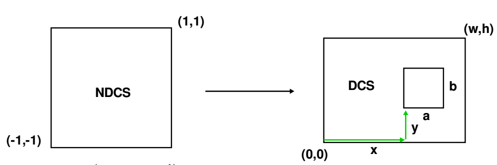
$$x_{NDCS} = -1/z_{VCS}$$
$$y_{NDCS} = 1/z_{VCS}$$
$$z_{NDCS} = \frac{5}{3} + \frac{8}{3z_{VCS}}$$

Viewport Transformation

- generate pixel coordinates
  - map x, y from range –1…1 (*normalized device coordinates*) to pixel coordinates on the display
  - involves 2D scaling and translation



Viewport Transformation



$$x_{DCS} = w\frac{(x_{NDCS}+1)}{2}$$
$$y_{DCS} = h\frac{(y_{NDCS}+1)}{2}$$
$$z_{DCS} = \frac{(z_{NDCS}+1)}{2}$$

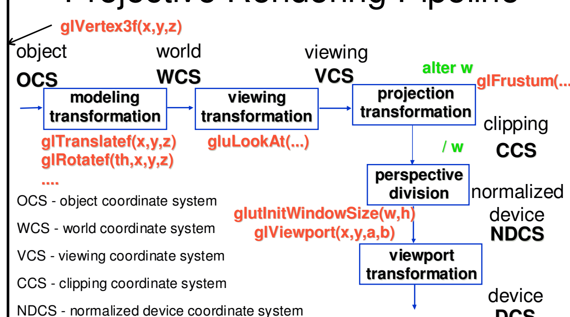
OpenGL

```
glViewport(x,y,a,b);
```

default:

```
glViewport(0,0,w,h);
```


Projective Rendering Pipeline



OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

University of British Columbia

CPSC 414 Computer Graphics

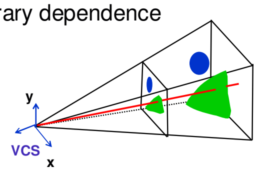
Picking

Interactive Object Selection

- move cursor over object, click
  - how to decide what is below?
- ambiguity
  - many 3D world objects map to same 2D point
- four common approaches
  - manual ray intersection
  - bounding extents
  - backbuffer color coding
  - selection region with hit list

Manual Ray Intersection

- do all computation at application level
  - map selection point to a ray
  - intersect ray with all objects in scene.
- advantages
  - no library dependence

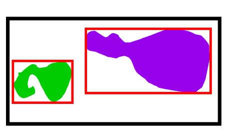


Manual Ray Intersection

- do all computation at application level
  - map selection point to a ray
  - intersect ray with all objects in scene.
- advantages
  - no library dependence
- disadvantages
  - difficult to program
  - slow: work to do depends on total number and complexity of objects in scene

Bounding Extents

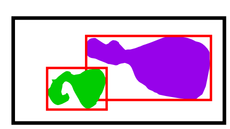
- keep track of axis-aligned bounding rectangles



- advantages
  - conceptually simple
  - easy to keep track of boxes in world space

Bounding Extents

- disadvantages
  - low precision
  - must keep track of object-rectangle relationship
- extensions
  - do more sophisticated bound bookkeeping

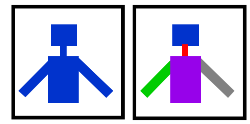


Backbuffer Color Coding

- use backbuffer for picking
  - create image as computational entity
  - never displayed to user
- redraw all objects in backbuffer
  - turn off shading calculations
  - set unique color for each pickable object
    - store in table
  - read back pixel at cursor location
    - check against table

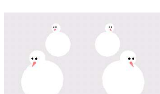
Backbuffer Color Coding

- advantages
  - conceptually simple
  - variable precision
- disadvantages
  - number of color bits must be adequate
  - introduce 2x redraw delay

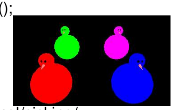


Backbuffer Example

```
glColor3f(1.0f, 1.0f, 1.0f);
for(int i = 0; i < 2; i++)
    for(int j = 0; j < 2; j++) {
        glPushMatrix();
        glTranslatef(i*3.0,0,-j * 3.0);
        glColor3f(1.0f, 1.0f, 1.0f);
        glCallList(snowman_display_list);
        glPopMatrix();
    }
```



```
for(int i = 0; i < 2; i++)
  for(int j = 0; j < 2; j++) {
    glPushMatrix();
    switch (i*2+j) {
      case 0: glColor3ub(255,0,0);break;
      case 1: glColor3ub(0,255,0);break;
      case 2: glColor3ub(0,0,255);break;
      case 3: glColor3ub(250,0,250);break;
    }
    glTranslatef(i*3.0,0,-j * 3.0)
    glCallList(snowman_display_list);
    glPopMatrix();
  }
```



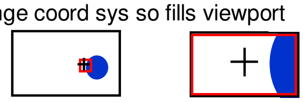
http://www.lighthouse3d.com/opengl/picking/

Select/Hit

- use small region around cursor for viewport
- assign per-object integer keys (names)
- redraw in special mode
- store hit list of objects in region
- examine hit list
- OpenGL support

Viewport

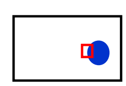
- small rectangle around cursor
  - change coord sys so fills viewport



- why rectangle instead of point?
  - people aren't great at positioning mouse
    - Fitts's Law: time to acquire a target is function of the distance to and size of the target
  - allow several pixels of slop

Viewport

- tricky to compute
  - invert viewport matrix, set up new orthogonal projection
- simple utility command
  - gluPickMatrix(x,y,w,h,viewport)
    - x,y: cursor point
    - w,h: sensitivity/slop (in pixels)
  - push old setup first, so can pop it later



Render Modes

- glRenderMode(mode)
  - GL_RENDER: normal color buffer
    - default
  - GL_SELECT: selection mode for picking
  - (GL_FEEDBACK: report objects drawn)

Name Stack

- “names” are just integers

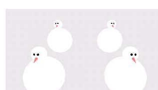
  glInitNames()
- flat list

  glLoadName(name)
- or hierarchy supported by stack

  glPushName(name), glPopName
  - can have multiple names per object

Hierarchical Names Example

```
for(int i = 0; i < 2; i++) {
  glPushName(i);
  for(int j = 0; j < 2; j++) {
    glPushMatrix();
    glPushName(j);
    glTranslatef(i*10.0,0,j * 10.0);
      glPushName(HEAD);
      glCallList(snowManHeadDL);
      glLoadName(BODY);
      glCallList(snowManBodyDL);
      glPopName();
    glPopName();
    glPopMatrix();
  }
  glPopName();
}
```



http://www.lighthouse3d.com/opengl/picking/

Hit List

- glSelectBuffer(buffersize, *buffer)
  - where to store hit list data
- on hit, copy entire contents of name stack to output buffer.
- hit record
  - number of names on stack
  - minimum and minimum depth of object vertices
    - depth lies in the z-buffer range [0,1]
    - multiplied by 2^32 -1 then rounded to nearest int

Separate Pick Function?

- use same function to draw and pick
  - simpler to code
  - name stack commands ignored in render mode
- customize functions for each
  - potentially more efficient
  - can avoid drawing unpickable objects

Select/Hit

- advantages
  - faster
    - OpenGL support means hardware accel
    - only do clipping work, no shading or rasterization
  - flexible precision
    - size of region controllable
  - flexible architecture
    - custom code possible, e.g. guaranteed frame rate
- disadvantages
  - more complex