University of British Columbia

CPSC 414 Computer Graphics

Viewing and Projections

Wed 17 Sep 2003

- project 1
- recap: display lists
- viewing
- projections

News

- Project 1 out
- Trouble ticket into IT services re newsgroup on news.interchange
  - read on nnrp.cs in the meantime

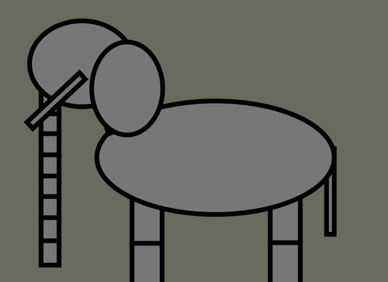
Project 1: Articulated Elephant

- modelling
  - spheres and cubes
  - hierarchical transformations
  - think cartoon!
- animation
  - more transformations
  - tail wag, head/neck nod, leg raise, trunk curl
  - gaps, self-intersections OK

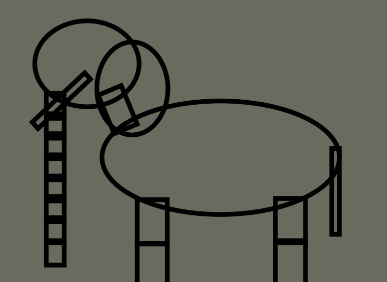
Elephants

http://www.wapers.com/animalnplant/elephant/elephant.jpg
http://www.worldwildlife.org/expeditions/images/elephant.jpg
http://aletsch.esis.ch/julen/picts/elephant.jpg

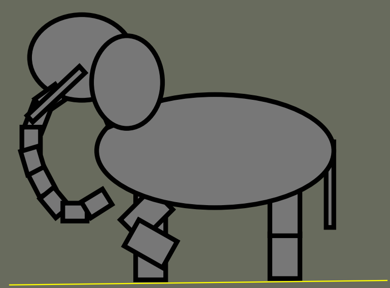
Elephant Rest Pose



Elephant Structure



Trunk Curled, Leg Raised



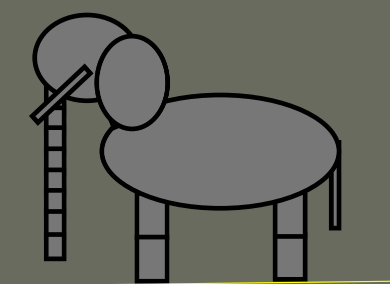
Interaction

- key bindings as toggles
- on click, move from rest state to new position or vice versa
- already in framework: 6 camera positions
- toggle between jumpcut and smooth transition

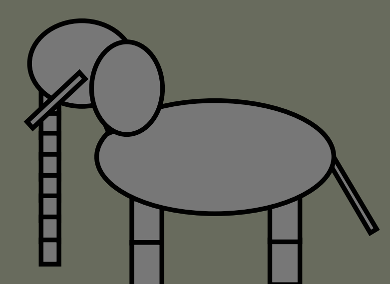
Transition

- first: jump cut from old to new position
  - all change happens in single frame
- do last: add smooth transition
  - change happens gradually over 30 frames
  - key click triggers animation loop
    - explicitly redraw 30 times
    - linear interpolation:
      each time, param += (new-old)/30
  - example: 5-frame transition

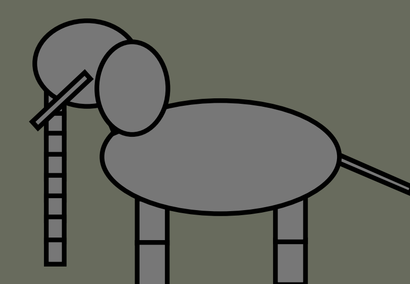
Tail Wag Frame 0



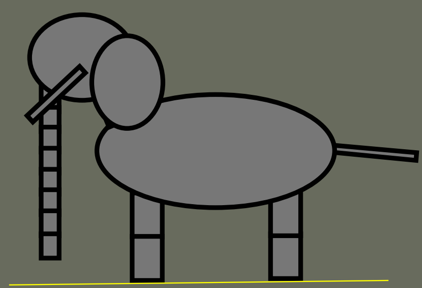
Tail Wag Frame 1



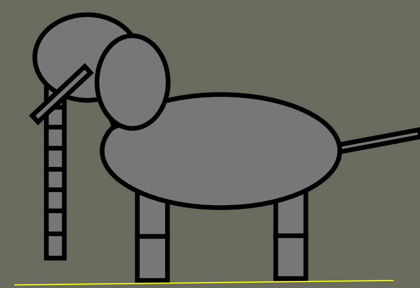
Tail Wag Frame 2



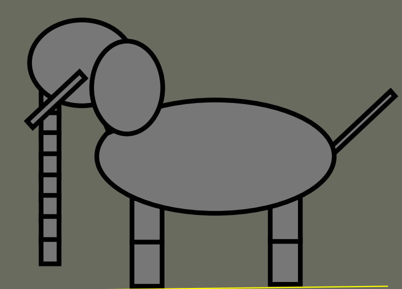
Tail Wag Frame 3



Tail Wag Frame 4



Tail Wag Frame 5



Strategy

- check from all camera angles
- interleave modelling, animation
  - add body part, then animate it
  - discover if on wrong track sooner
  - depenencies: can't get anim credit if no model
- do smooth transitions last
- don't start extra credit until required all done
- consider using different model, anim xforms

Writeup

- README
  - what's implemented
  - undone: for partial credit
    - state problems
    - describe how far you got
    - conjecture possible solutions
  - extra
    - what you did
    - how many points you argue it's worth

Grading

- Project 1: 10% of course grade
- use handin program before Thu 2 Oct 5pm
- face-to-face grading
  - sign up for 10-minute slot, arrive 10 min early
  - bring printouts: code, README
    - must match handin
  - demo from submission directory
  - late if handin or file timestamps after deadline
    - late policy: 3 grace days for term, then 20% per day

Plagarism Policy

- no collaboration allowed
  - your work alone
  - general discussions of approach OK
  - do not look at (or copy) anybody else's code
- plagarism is detectable
  - both by TAs and automated programs

Hall of Fame

- best work posted on course web site
- previous years
  - http://www.ugrad.cs.ubc.ca/~cs414/Vjan2003/best_projects
  - http://www.ugrad.cs.ubc.ca/~cs414/best_of_2002/HW3_best.htm

Display List recap

- reuse block of OpenGL code
- more efficient than immediate mode
  - code reuse, driver optimization
- good for static objects redrawn often
  - can't change contents
  - not just for multiple instances
    - interactive graphics: objects redrawn every frame

Display List recap

- example: 36 snowmen
  - small display list with 36x reuse
    - 3x faster
  - big display list with 1x reuse
    - 2x faster
  - nested display lists, 1x * 36x reuse:
    - 3x faster, high-level block available

Double Buffering recap

- two framebuffers, front and back
  - avoid flicker
  - while front is on display, draw into back
  - when drawing finished, swap the two

University of British Columbia

CPSC 414 Computer Graphics

Viewing and Projections

Viewing and Projection

- need to get from 3D world to 2D image
- projection: geometric abstraction
  - what eyes or cameras do
- two pieces
  - viewing transform:
    - where is the camera, what is it pointing at?
  - perspective transform: 3D -> 2D
    - flatten to image

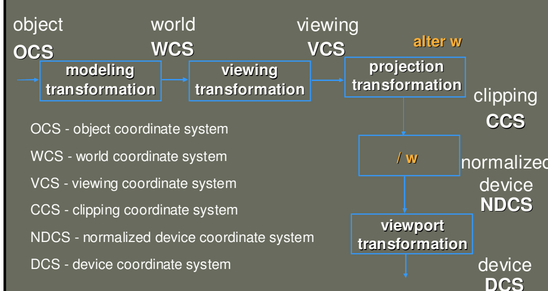
From Geometry to Screen

- geometry in world coordinate system: how to get to screen?
  - transform to camera coordinate system
  - transform to volume in viewing coordinates
  - clip
  - project to display coordinates
  - rasterize

Coordinate Systems

- result of a transformation
- names
  - convenience
    - elephant: neck, head, tail
  - standard conventions in graphics pipeline
    - object/modelling
    - world
    - camera/viewing
    - screen/window
    - raster/device

Projective Rendering Pipeline

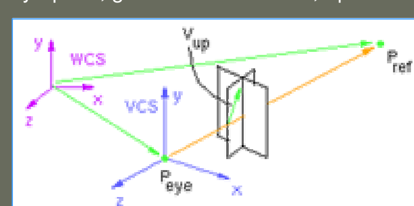


Basic Viewing

- starting spot - OpenGL
  - camera at world origin
    - probably inside an object
  - y axis is up
  - looking down negative z axis
    - why? RHS with x horizontal, y vertical
- translate backward so scene is visible
  - move distance d = focal length
- what about flying around?

Arbitrary Viewing Position

- rotate/translate/scale not intuitive
- convenient formulation
  - eye point, gaze/lookat direction, up vector

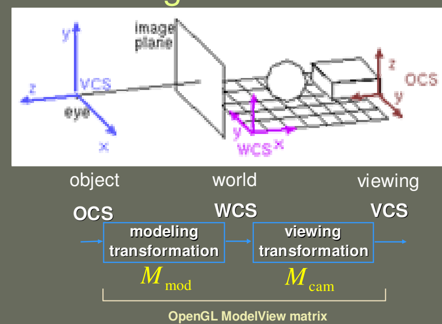


Viewing Transformation

- translate *eye* to origin
- rotate view vector (*lookat – eye*) to z axis
- rotate around z to bring *up* into yz-plane

M_{cam}

Viewing Transformation



Viewing Transformation

- OpenGL
  - `gluLookAt(ex,ey,ez,lx,ly,lz,ux,uy,uz)`

  but this postmultiplies the current matrix; therefore usually use as follows:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(ex,ey,ez,lx,ly,lz,ux,uy,uz)
// now ok to do model transformations
```

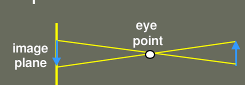

University of British Columbia

CPSC 414 Computer Graphics

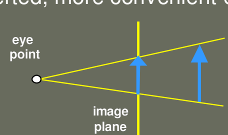
Projections

Projection

- theoretical pinhole camera

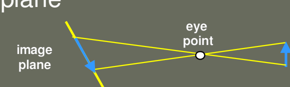


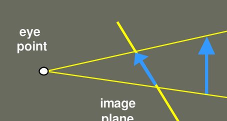
– image inverted, more convenient equivalent



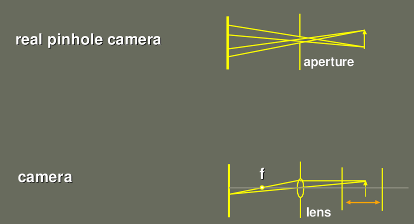
General Projection

image plane need not be perpendicular to view plane



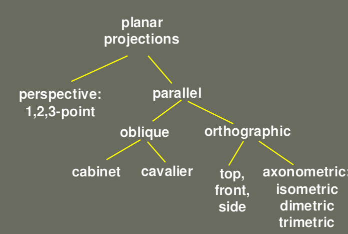


Real Cameras



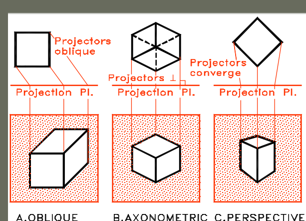
price to pay: limited depth of field

Projection Taxonomy



Projection Comparison

- Obliques
  - Cavalier
  - Cabinet
- Axonometrics
  - Isometrics
  - Others
- Perspectives

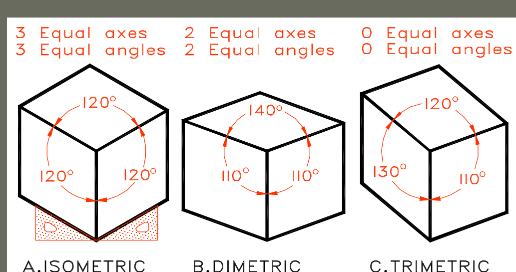


http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20

Oblique Projections

- both have true front view
  - cavalier: distance true
  - cabinet: distance half

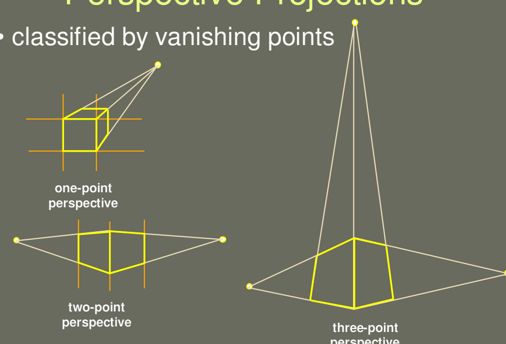
Axonometric Projections



http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20

Perspective Projections

- classified by vanishing points



Projective Transformations

- planar geometric projections
  - planar: onto a plane
  - geometric: using straight lines
  - projections: 3D -> 2D
- aka projective mappings
- counterexamples?

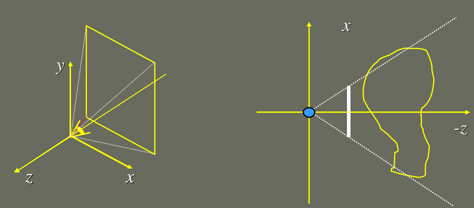
Projective Transformations

- properties
  - lines mapped to lines and triangles to triangles
  - parallel lines do NOT remain parallel
    - e.g. rails vanishing at infinity
  - affine combinations are NOT preserved
    - e.g. center of a line does not map to center of projected line (perspective foreshortening)

Perspective Projection

- project all geometry

through a common ***center of projection*** (***eye point***)
onto an ***image plane***

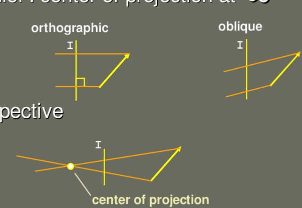


Projection

definition
mapping $f:\Re^n \rightarrow \Re^m,\ m<n$

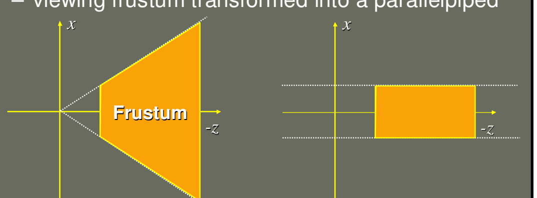
parallel : center of projection at ∞

perspective



Projective Transformations

- transformation of space
  - center of projection moves to infinity
  - viewing frustum transformed into a parallelpiped

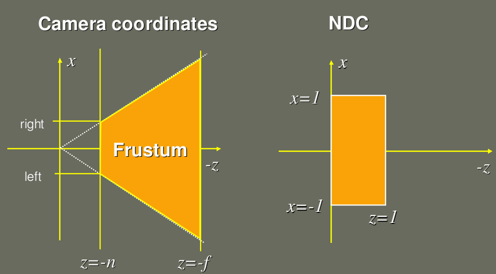


View Volume

- convention
  - viewing frustum mapped to specific parallelpiped
    - Normalized Device Coordinates (NDC)
  - only objects inside the parallelpiped get rendered
  - which parallelpied? depends on rendering system
- OpenGL
  - left/right image boundaries mapped to $x = +/- 1$
  - top/bottom mapped to $y = +/- 1$
  - near/far plane mapped to $z = 0,\ z = 1$

Projective Transformations

- OpenGL convention

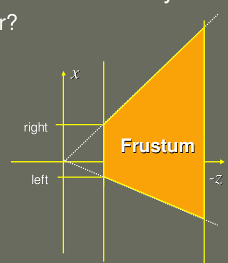


Projective Transformations

- why near and far plane?
  - near plane:
    - avoid singularity (division by zero, or very small numbers)
  - far plane:
    - store depth in fixed-point representation (integer), thus have to have fixed range of values (0…1)
    - avoid/reduce numerical precision artifacts for distant objects

Asymmetric Frusta

- our formulation allows asymmetry
  - why bother?



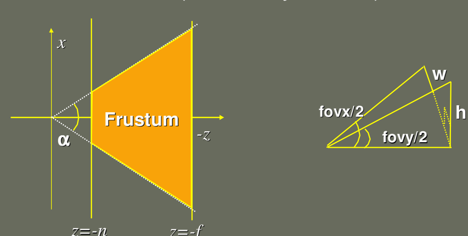
Simpler Formulation

- look through window center
  - symmetric frustum
- left, right, bottom, top, near, far
  - overkill
  - nonintuitive
- constraints

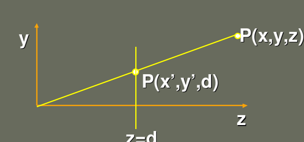
  left = -right, bottom = -top

Field-of-View Formulation

- FOV in one direction + aspect ratio (w/h)
  - determines FOV in other direction
  - also set near, far (reasonably intuitive)



Basic Projection



similar triangles: $\frac{y'}{d} = \frac{y}{z} \qquad y' = \frac{y \cdot d}{z}$

similarly $x' = \frac{x \cdot d}{z}$

Basic Projection

- using w and 4x4 matrices

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

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

Projective Transformations

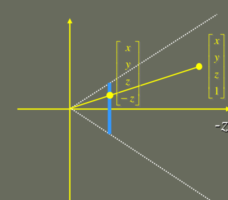
- can express as homogeneous 4x4 matrices!
- 16 matrix entries
  - multiples of the same matrix all describe the same transformation
  - 15 degrees of freedom
  - mapping of 5 points uniquely determines transformation

Projective Transformations

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

Perspective Projection

- example
  - Assume image plane at $z = -1$
  - A point $[x,y,z,1]^T$ projects to $[-x/z, -y/z, -z/z, 1]^T \equiv [x, y, z, -z]^T$



Perspective Projection

$$T\left(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -z \end{bmatrix} \equiv \begin{bmatrix} -x/z \\ -y/z \\ -1 \\ 1 \end{bmatrix}$$