University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2016

Tamara Munzner

Viewing 4

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2016

Projective Rendering Pipeline

object OCS —O2W→ world WCS —W2V→ viewing VCS —V2C→ clipping CCS

modeling transformation → viewing transformation → projection transformation → perspective divide (C2N) → normalized device NDCS → viewport transformation (N2D) → device DCS

OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

CCS - clipping coordinate system

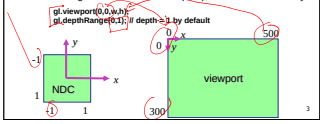
NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

NDC to Device Transformation

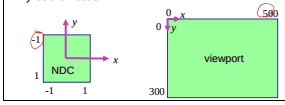
- map from NDC to pixel coordinates on display
  - NDC range is x = -1...1, y = -1...1, z = -1...1
  - typical display range: x = 0...500, y = 0...300
    - maximum is size of actual screen
    - z range max and default is (0, 1), use later for visibility

gl.viewport(0,0,w,h);
gl.depthRange(0,1); // depth = 1 by default



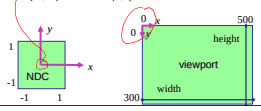
Origin Location

- yet more (possibly confusing) conventions
  - GL origin: lower left
  - most window systems origin: upper left
- then must reflect in y
- when interpreting mouse position, have to flip your y coordinates



N2D Transformation

- general formulation
  - reflect in y for upper vs. lower left origin
  - scale by width, height, depth
  - translate by width/2, height/2, depth/2
    - FCG includes additional translation for pixel centers at (.5, .5) instead of (0,0)

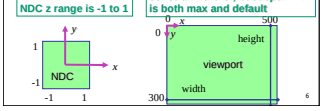


N2D Transformation

$$\begin{bmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & \frac{depth}{2} & \frac{depth}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{NDCS} \\ y_{NDCS} \\ z_{NDCS} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{width}{2}(x_{NDCS}+1)-1 \\ \frac{height}{2}(-y_{NDCS}+1)-1 \\ \frac{depth}{2}(z_{NDCS}+1) \\ 1 \end{bmatrix}$$

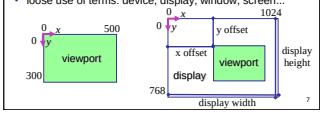
reminder:
NDC z range is -1 to 1

Display z range is 0 to 1.
gl.depthRange(n,f) can constrain further, but *depth* = 1 is both max and default



Device vs. Screen Coordinates

- viewport/window location wrt actual display not available within GL
  - usually don't care
    - use relative information when handling mouse events, not absolute coordinates
  - could get actual display height/width, window offsets from OS
- loose use of terms: device, display, window, screen...



Projective Rendering Pipeline

vertex(x,y,z)

object OCS —O2W→ world WCS —W2V→ viewing VCS —V2C→ clipping CCS

modeling transformation (translatef(x,y,z), rotatef(a,x,y,z), ...) → viewing transformation (lookAt(...)) → projection transformation (alter w; frustum(...)) → perspective division (C2N, / w) → normalized device NDCS → viewport transformation (N2D) → device DCS

canvas.{w,h}
viewport(x,y,a,b)

OCS - object coordinate system

WCS - world coordinate system

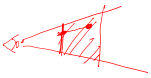
VCS - viewing coordinate system

CCS - clipping coordinate system

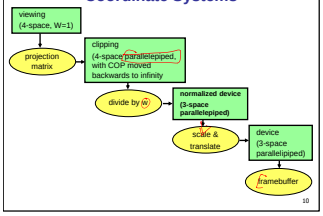
NDCS - normalized device coordinate system

DCS - device coordinate system

Questions?



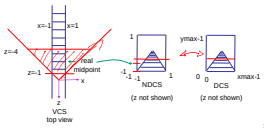
Coordinate Systems



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

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

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1 & -1 \\ & & -5/3 & -8/3 \end{bmatrix} \begin{bmatrix} x_{VCS} \\ y_{VCS} \\ z_{VCS} \\ -z_{VCS} \end{bmatrix} = \begin{bmatrix} x_{VCS} \\ y_{VCS} \\ -z_{VCS} \\ 5z_{VCS}/3 - 8/3 \end{bmatrix} \quad / \text{ w}$$

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

Projective Rendering Pipeline

following pipeline from top/left to bottom/right: moving object POV

glVertex3f(x,y,z)

object OCS —O2W→ world WCS —W2V→ viewing VCS —V2C→ clipping CCS

modeling transformation (glTranslatef(x,y,z), glRotatef(a,x,y,z), ...) → viewing transformation (gluLookAt(...)) → projection transformation (alter w; glFrustum(...)) → perspective division (C2N, / w) → normalized device NDCS → viewport transformation (N2D) → device DCS

glutInitWindowSize(w,h)
glViewport(x,y,a,b)

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

OpenGL Example

go back from end of pipeline to beginning: coord frame POV!

object OCS —O2W→ world WCS —W2V→ viewing VCS —V2C→ clipping CCS

modeling transformation → viewing transformation → projection transformation

CCS: gl.viewport(0,0,w,h);

VCS: THREE.PerspectiveCamera(view angle, aspect, near, far)

WCS:
```
u_xformMatrix = Identidy()
gl.uniformMatrix4fv(u_xformMatrix, false, xformMatrix);
```

OCS1:
```
torsoGeometry.applyMatrix(u_xformMatrix );
var torso = new THREE.Mesh(torsoGeometry,normalMaterial);
scene.add(torso);
```


Coord Sys: Frame vs Point

read down: transforming between coordinate frames, from frame A to frame B

read up: transforming points, up from frame B coords to frame A coords

OpenGL command order

	DCS display		
D2N	gl.Viewport(x,y,a,b)		N2D
	NDCS normalized device		
N2V	glFrustum(...)		V2N
	VCS viewing		
V2W	gluLookAt(...)		W2V
	WCS world		
W2O	glRotatef(a,x,y,z)		O2W
	OCS object		
	glVertex3f(x,y,z)		

pipeline interpretation

Questions?

17