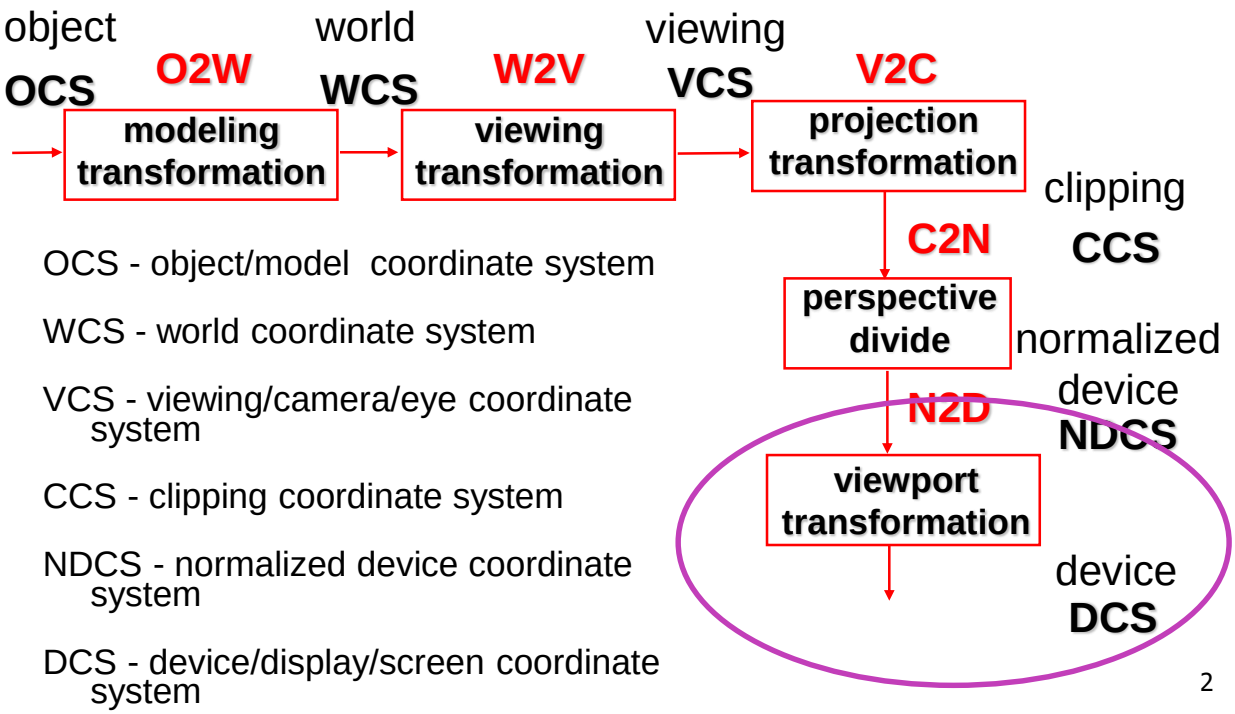
University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2016

Tamara Munzner

Viewing 4

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

Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

CCS - clipping coordinate system

NDCS - normalized device coordinate system

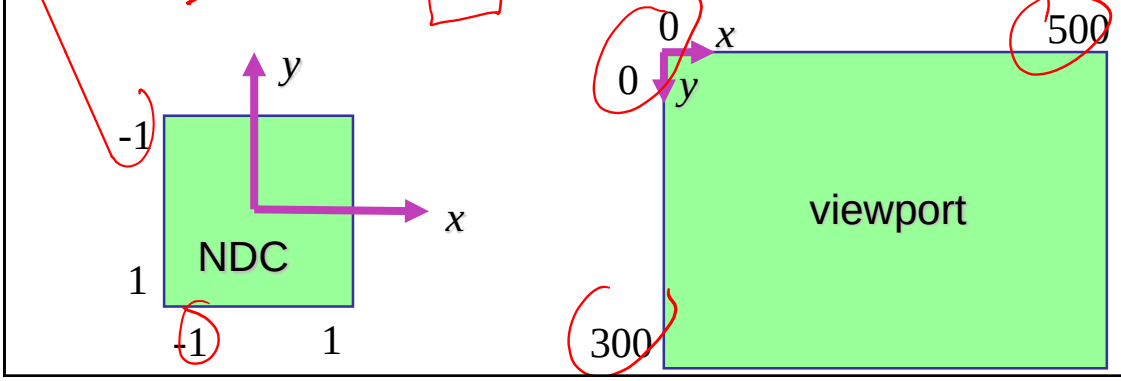
DCS - device/display/screen coordinate system

2

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

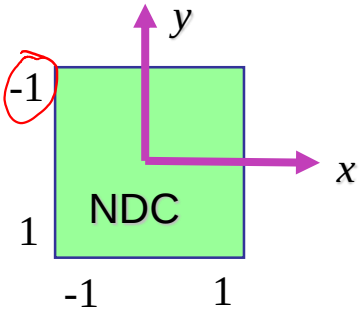
```
gl.viewport(0,0,w,h);
gl.depthRange(0,1); // depth = 1 by default
```

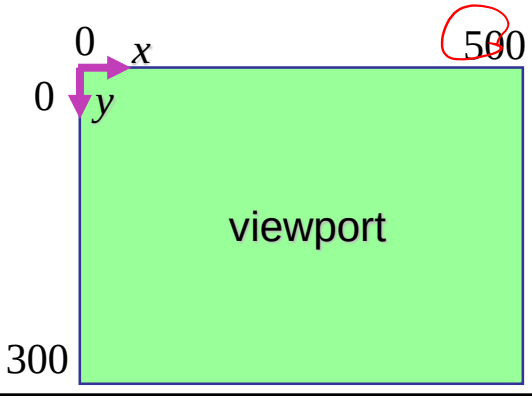


3

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

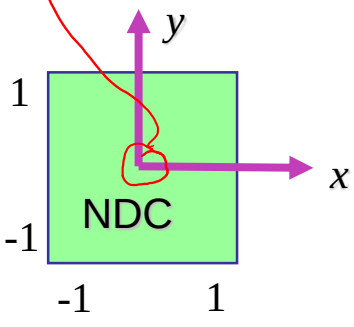


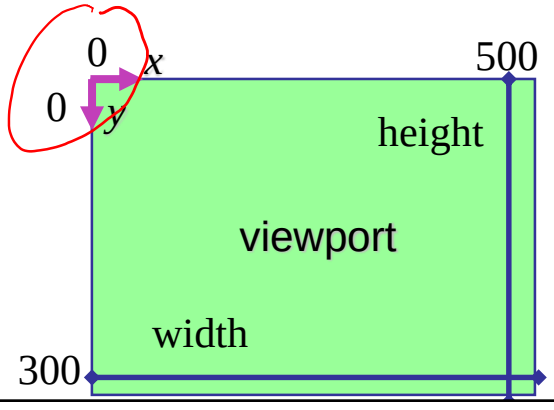


4

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)





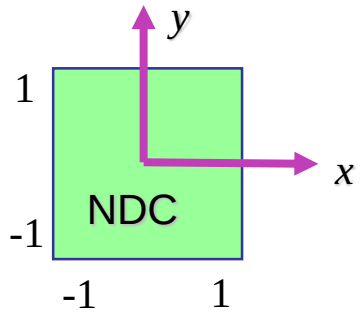
5

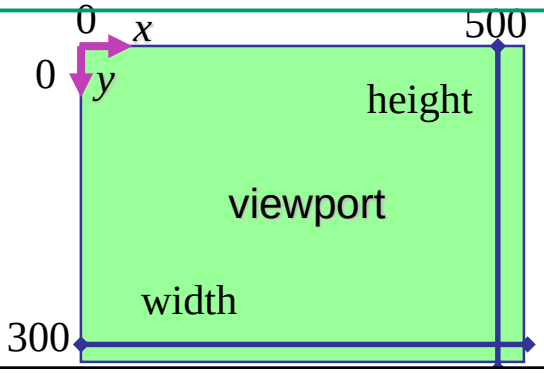
N2D Transformation

$$\begin{bmatrix} x_D \\ y_D \\ z_D \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \frac{width}{2} - 1 \\ 0 & 1 & 0 & \frac{height}{2} - 1 \\ 0 & 0 & 1 & \frac{depth}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{width}{2} & 0 & 0 & 0 \\ 0 & \frac{height}{2} & 0 & 0 \\ 0 & 0 & \frac{depth}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_N \\ y_N \\ z_N \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{width(x_N + 1)}{2} - 1 \\ \frac{height(-y_N + 1)}{2} - 1 \\ \frac{depth(z_N + 1)}{2} \\ 1 \end{bmatrix}$$

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

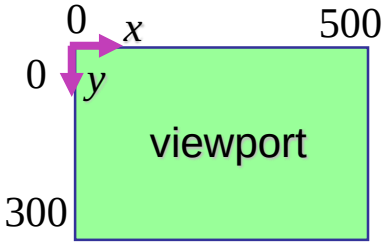


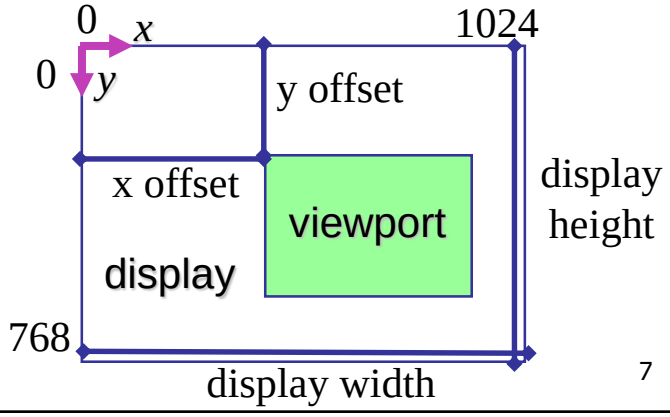


6

Device vs. Screen Coordinates

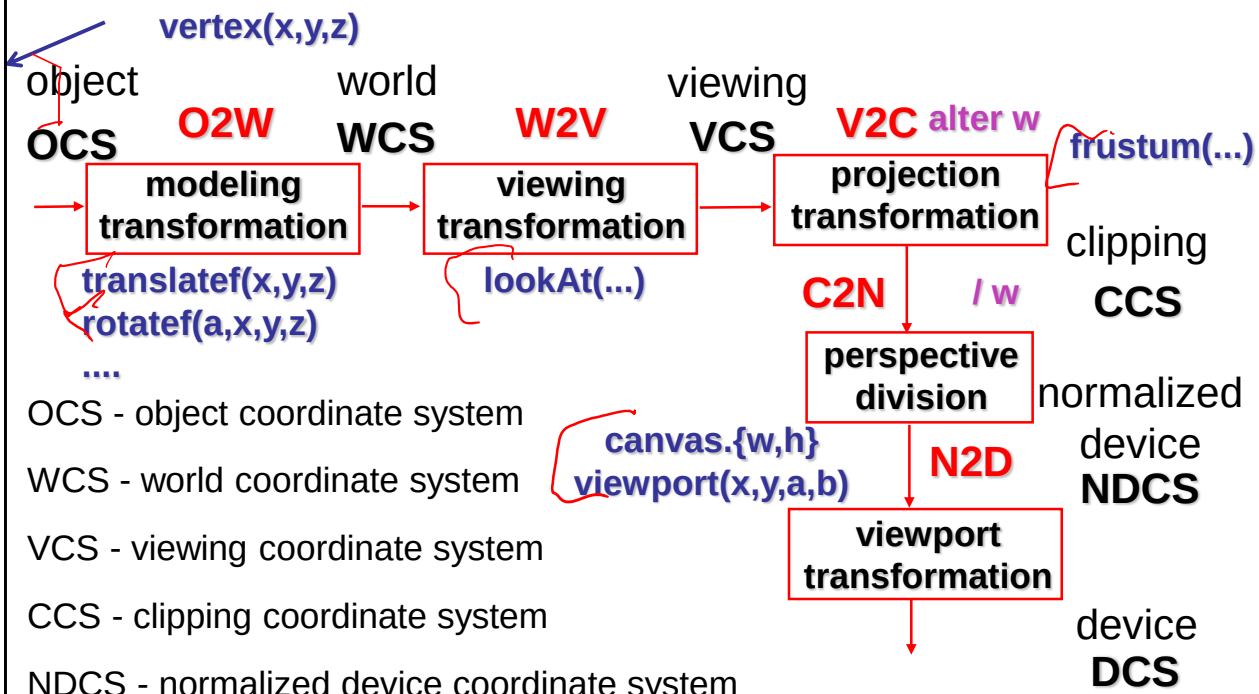
- viewport/window location wrt actual display not available within GL
  - usually don't care
    - use relative information when handling mouse events, not absolute coordinates
  - could get actual display height/width, window offsets from OS
- loose use of terms: device, display, window, screen...





7

Projective Rendering Pipeline



OCS - object coordinate system

WCS - world coordinate system

VCS - viewing coordinate system

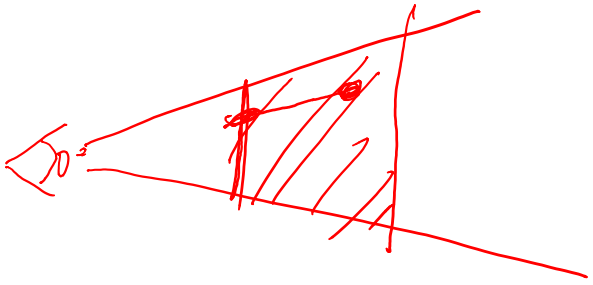
CCS - clipping coordinate system

NDCS - normalized device coordinate system

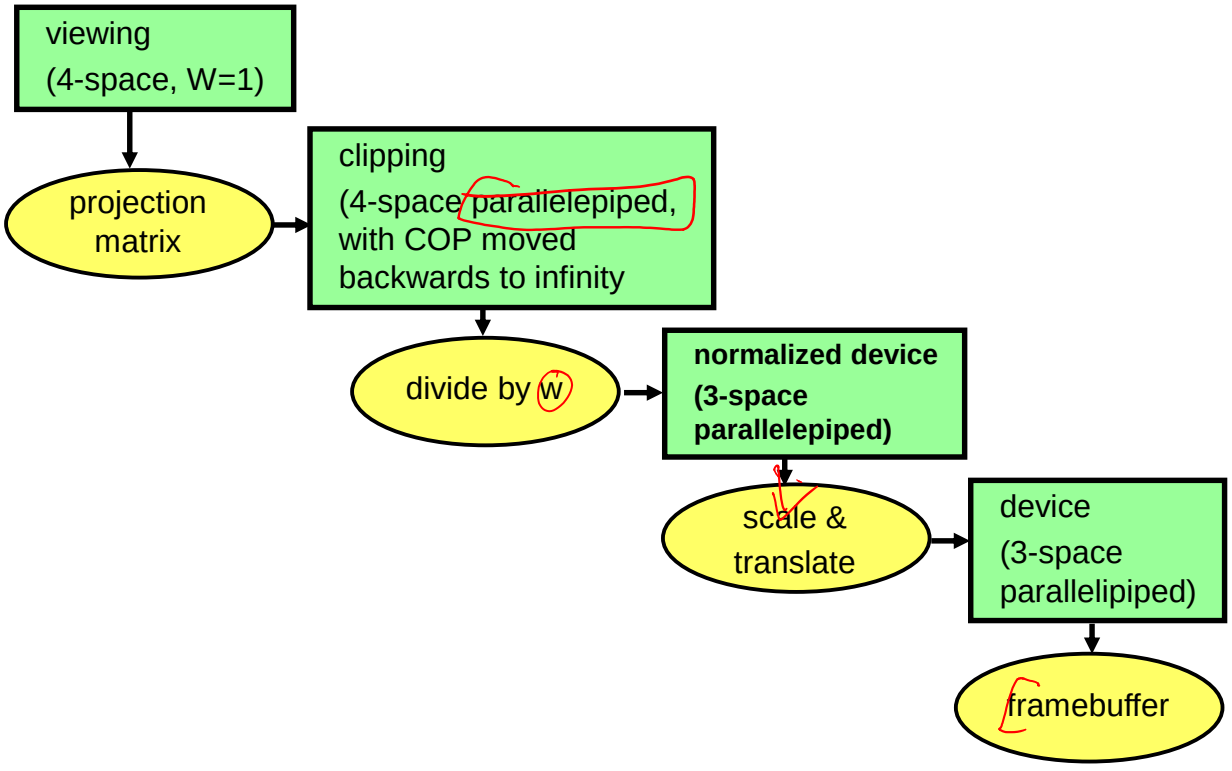
DCS - device coordinate system

8

Questions?



Coordinate Systems

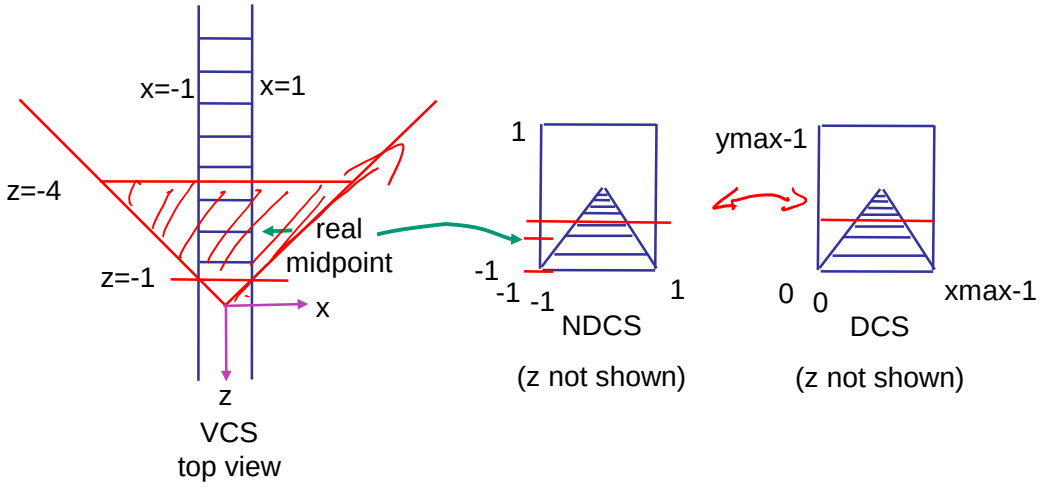


10

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



11

Perspective Example

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

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

12

Perspective Example

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -5z_{VCS}/3 & -8/3 & -5/3 & -8/3z_{VCS} \\ 0 & 0 & 0 & 1 \\ -z_{VCS} & 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

/ w

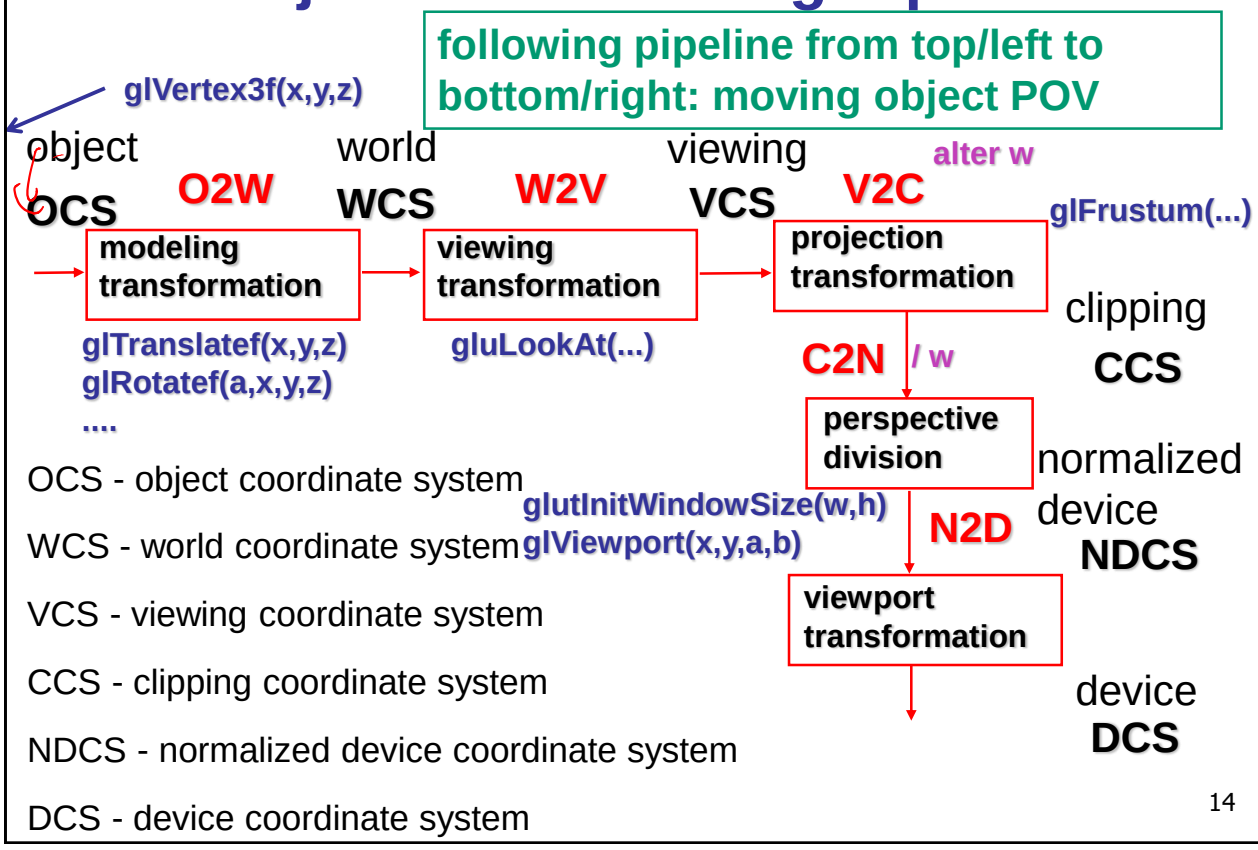
$$x_{NDCS} = -1/z_{VCS}$$

$$y_{NDCS} = 1/z_{VCS}$$

$$z_{NDCS} = \frac{5}{3} + \frac{8}{3z_{VCS}}$$

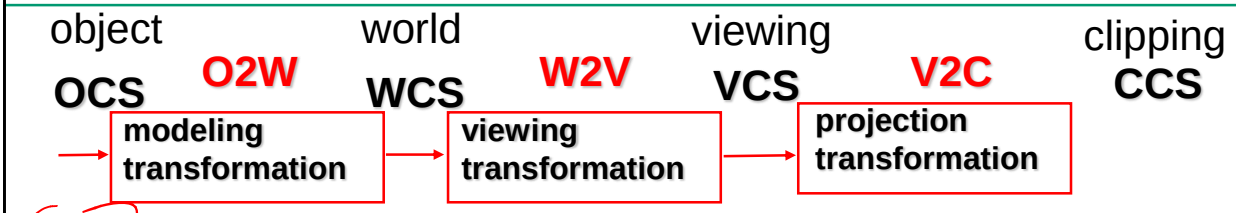
13

Projective Rendering Pipeline



OpenGL Example

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



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

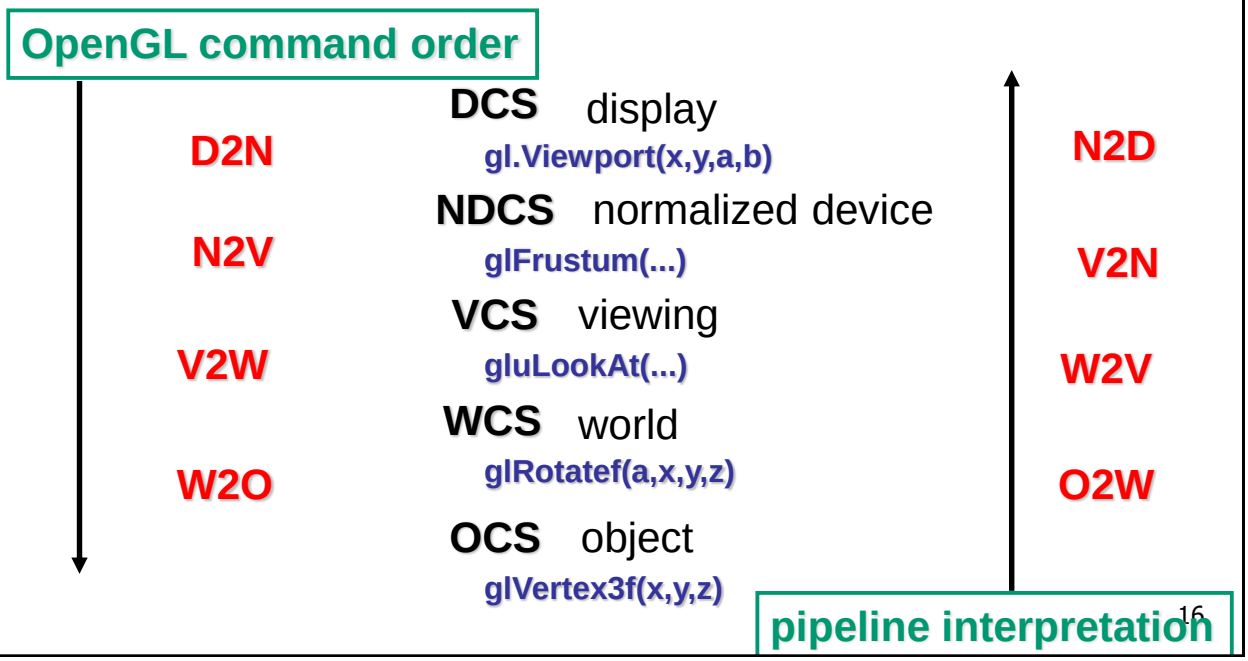
```
torsoGeometry.applyMatrix(u_xformMatrix );
var torso = new THREE.Mesh(torsoGeometry,normalMaterial);
scene.add(torso);
```

15

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



Questions?