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

Tamara Munzner

Viewing 4

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

modeling transformation  viewing transformation  projection transformation

O2W  W2V  V2C

perspective divide

C2N  N2D

viewport transformation

clipping
CCS
normalized
device
NDCS
device
DCS
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

```javascript
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

[Diagram showing NDC space and viewport with origin locations marked]
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

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\frac{1}{\text{width}} & 0 & 0 \\
0 & \frac{1}{\text{height}} & 0 \\
0 & 0 & \frac{1}{\text{depth}}
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & x_N \\
0 & 0 & 0 & 1 & 0 & 0 & y_N \\
0 & 0 & 0 & 0 & 1 & 0 & z_N \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
= \begin{bmatrix}
\frac{\text{width}(x_N + 1)}{2} & 0 & 0 & 0 & 0 & 0 & x_N \\
0 & \frac{\text{height}(y_N + 1)}{2} & 0 & 0 & 0 & 0 & y_N \\
0 & 0 & \frac{\text{depth}(z_N + 1)}{2} & 0 & 0 & 0 & z_N \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

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

Reminder: NDC z range is -1 to 1
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...

![Diagram showing the relationship between viewport and display coordinates.](image)
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

vertex(x,y,z)
object
O2W
modeling transformation

W2V
viewing transformation

V2C
projection transformation

C2N
perspective division

N2D
viewport transformation

frustum(...) clipping CCS

canvas.{w,h} normalized device NDCS
device DCS

viewport(x,y,a,b)

O2W W2V V2C C2N N2D

modeling transformation viewing transformation projection transformation

vertex(x,y,z) translatef(x,y,z) rotatef(a,x,y,z)...

lookAt(...)...
Questions?
Coordinate Systems

- Viewing (4-space, W=1)
- Projection matrix
- Clipping (4-space parallelepiped, with COP moved backwards to infinity)
- Divide by w
- Normalized device (3-space parallelepiped)
- Scale & translate
- Device (3-space parallelepiped)
- Framebuffer
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

VCS top view

real midpoint

NDCS (z not shown)

DCS (z not shown)
**Perspective Example**

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

<table>
<thead>
<tr>
<th></th>
<th>( \frac{2n}{r} )</th>
<th>0</th>
<th>( \frac{r + l}{l} )</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{2n}{t} )</td>
<td>0</td>
<td>( \frac{r}{l} )</td>
<td>( \frac{t}{l} )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( \frac{t + b}{b} )</td>
<td>( \frac{t}{b} )</td>
<td>0</td>
<td>0</td>
<td>5/3</td>
<td>8/3</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( \frac{(f + n)}{f} )</td>
<td>( \frac{2fn}{n} )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Perspective Example

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
5z_{VCS} & 8/3 & 5/3 & 8/3 & z_{VCS} \\
z_{VCS} & 1 & 1
\end{bmatrix}
\]

\[
x_{ND\,CS} = \frac{1}{z_{VCS}} \\
y_{ND\,CS} = \frac{1}{z_{VCS}} \\
z_{ND\,CS} = \frac{5}{3} + \frac{8}{3z_{VCS}}
\]
Projective Rendering Pipeline

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

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

modeling transformation
viewing transformation
projection transformation
viewport transformation

perspective division

glVertex3f(x,y,z)
glTranslatef(x,y,z)
glRotatef(a,x,y,z)
....
gluLookAt(...)
glFrustum(...)
OpenGL Example

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

```
object OCS
  modeling transformation
  gl.viewport(0,0,w,h);

world WCS
  viewing transformation
  THREE.PerspectiveCamera(view angle, aspect, near, far)

viewing VCS
  projection transformation

clipping CCS

O2W

W2V

V2C

OCS

WCS

VCS

CCS

u_xformMatrix = Identity()
gl.uniformMatrix4fv(u_xformMatrix, false, xformMatrix);

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

D2N
N2V
V2W
W2O

DCS display
gl.Viewport(x,y,a,b)

NDCS normalized device
glOrtho(...)

VCS viewing
gluLookAt(...)

WCS world
glRotatef(a,x,y,z)

OCS object
glVertex3f(x,y,z)

pipeline interpretation
Questions?