DEPARTMENT OF COMPUTER GRAPHICS AND INTERACTION

APG Homework Assignment I

Jakub Hendrich, Daniel Meister

Library Management

- Library initialization
- Library finalization
  - Memory leaks etc. will be tested
- Context management
  - Context container
    - Creation, deletion
  - Current context

Context

- Color buffer
- Vertex buffer
  - Point size
  - Current color
- Transformation stack
- …

Color Buffer

- Image represented as 1D array of pixels
  - Use macro (or function) to map pixel coordinates to array index
    - e.g., `pixel2Index(x, y) = y*width + x`
- Pixel format
  - RGB float
- Clear and clear color
  - `sglClear` sets all pixels to the color specified by `sglClearColor`
    - If the `SGL_COLOR_BUFFER_BIT` bit is set
- Color buffer pointer
  - `sglGetColorBufferPointer` returns pointer to the red channel of the first pixel

Vertex Buffer

- Begin
  - `sglBegin` specifies the desired element type
    - `SGL_POINTS, SGL_LINES, SGL_LINE_STRIP, SGL_LINE_LOOP`
- Vertices insertion
  - `sglVertex2f, sglVertex3f, sglVertex4f`
    - Must be called between `sglBegin` and `sglEnd!`
- End
  - `sglEnd` transforms vertices using current model-view and projection matrix
  - `sglEnd` rasterizes the current element type using transformed vertices and the color specified by `sglColor3f`
  - `sglEnd` clears the buffer

Homogeneous Coordinates

- Points in $\mathbb{E}^n$ represented as vectors in $\mathbb{R}^{n+1}$
- Projection and affine transforms represented as homogeneous matrix i.e. square matrix in $\mathbb{R}^{n+1,n+1}$
- Composition of transformations represented as matrix multiplication
- Homogeneous coordinates of points in 3D $w \neq 0$

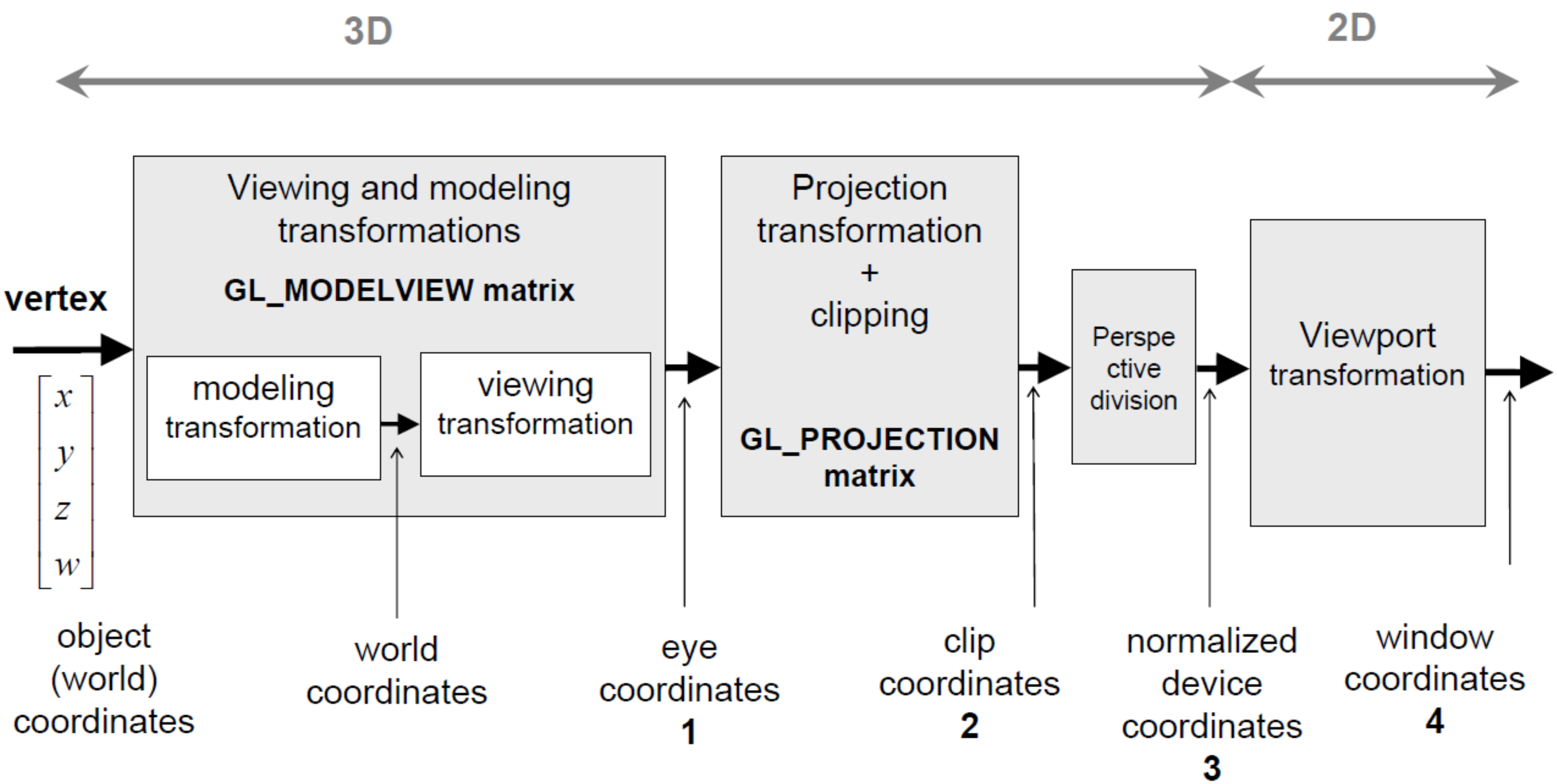
$$\underline{\mathbf{x}} \simeq [x, y, z, w]^\top \longrightarrow \mathbf{x} = [\tfrac{x}{w}, \tfrac{y}{w}, \tfrac{z}{w}]^\top$$

$$\mathbf{x} = [x, y, z]^\top \longrightarrow \underline{\mathbf{x}} \simeq [xw, yw, zw, w]^\top$$

- Point at infinity i.e. $w = 0$

$$\underline{\mathbf{x}} \simeq [x, y, z, 0]^\top$$

OpenGL Transformation Pipeline



Courtesy of Sloup and Felkel

Affine Transformation

- Translation and Scale

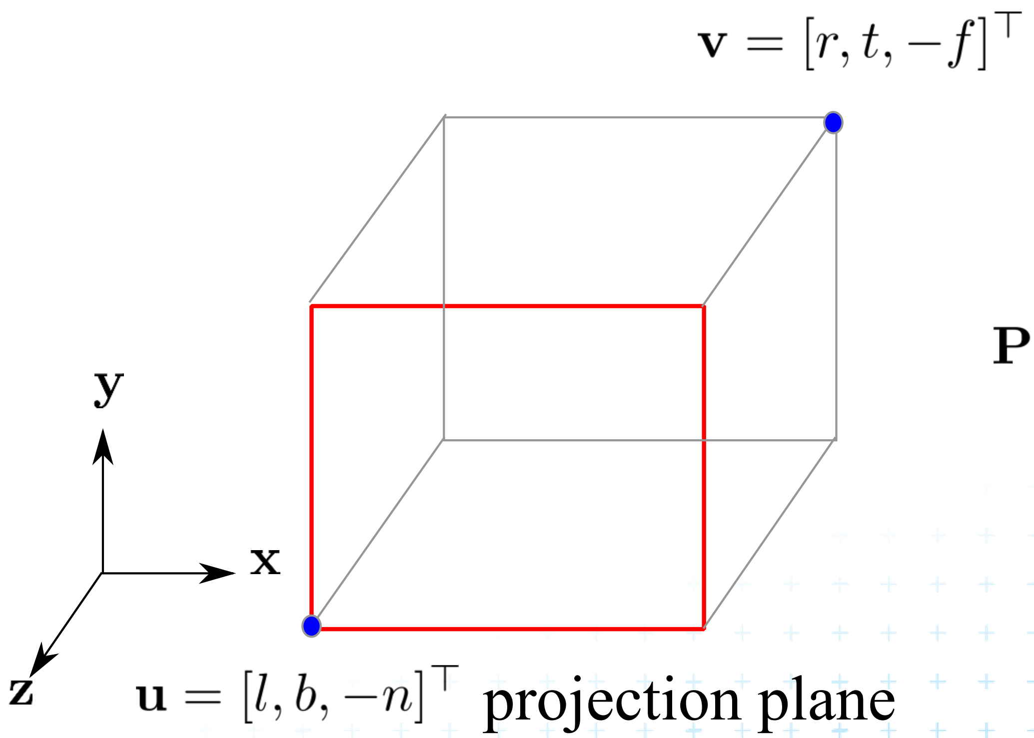
$$\mathbf{T}(\mathbf{t}) = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{S}(\mathbf{s}) = \begin{bmatrix} \mathbf{D}(\mathbf{s}) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation

$$\mathbf{R_y}(\varphi) = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R_z}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic Projection

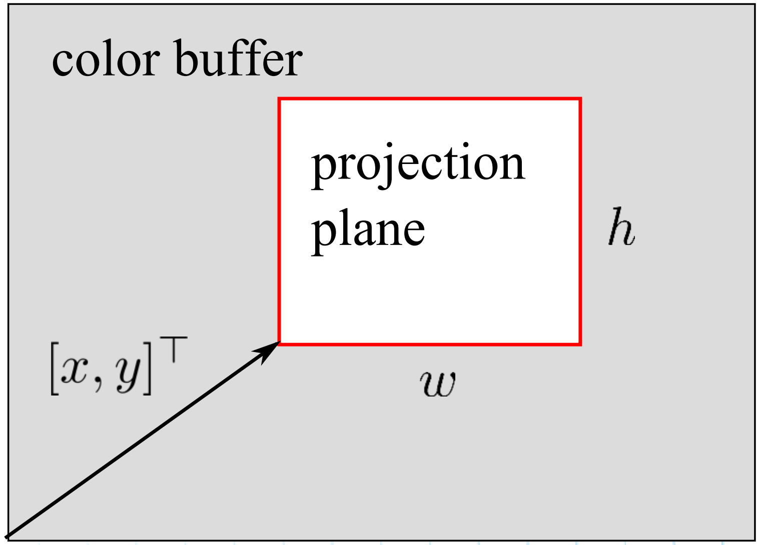
- Specifies parallel viewing volume
- Transformation from the viewing volume to $\langle -1, 1 \rangle^3$



$$\mathbf{P}_{\parallel} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Division and Viewport Trans.

- Perspective division
  - Maps homogeneous coordinates to Cartesian coordinates
  - Nonlinear operation
- Viewport Transformation
  - An affine transformation from projection plane to the color buffer



$$\mathbf{V} = \begin{bmatrix} \frac{w}{2} & 0 & x + \frac{w}{2} \\ 0 & \frac{h}{2} & y + \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation Stack

- Two modes
  - `SGL_MODELVIEW`, `SGL_PROJECTION`
    - `sglMatrixMode` sets the active mode
  - Use two separate buffers
- The current matrix is the top of the stack
- Matrices are column-major ordered!
  - `sglLoadMatrix`, `sglMultMatrix`

Transformation Stack

- Operations
  - `sglPushMatrix` duplicates the current matrix on the stack.
  - `sglPopMatrix` removes the top matrix from the stack

  - `sglLoadIdentity` replaces the current matrix by the identity matrix
  - `sglLoadMatrix` replaces the current matrix by a given matrix

  - `sglMultMatrix` multiplies the current matrix by a given matrix
  - `sglTranslate` multiplies the current matrix by translation matrix $\mathbf{T}(\mathbf{t})$
  - `sglScale` multiplies the current matrix by scale matrix $\mathbf{S}(\mathbf{s})$
  - `sglRotateY` multiplies the current matrix by rotation matrix $\mathbf{R}_\mathbf{y}(\varphi)$
  - `sglRotate2D` multiplies the current matrix by matrix $\mathbf{T}(\mathbf{c})\mathbf{R}_\mathbf{z}(\varphi)\mathbf{T}(\mathbf{c})^{-1}$
  - `sglOrtho` multiplies the current matrix by orthographic matrix $\mathbf{P}_\parallel$

Rasterization

- Line
  - Bresenham's algorithm for line
- Circle
  - Bresenham's algorithm for circle
- Ellipse and Arc
  - Approximation by 40 line segments (for full arc)
  - 1 bonus point: Adaptive approximation

Thank you for your attention!

Jakub Hendrich

30.9.2024