Tutorial 2

TA: Mengyun Liu, Hongtu Xu

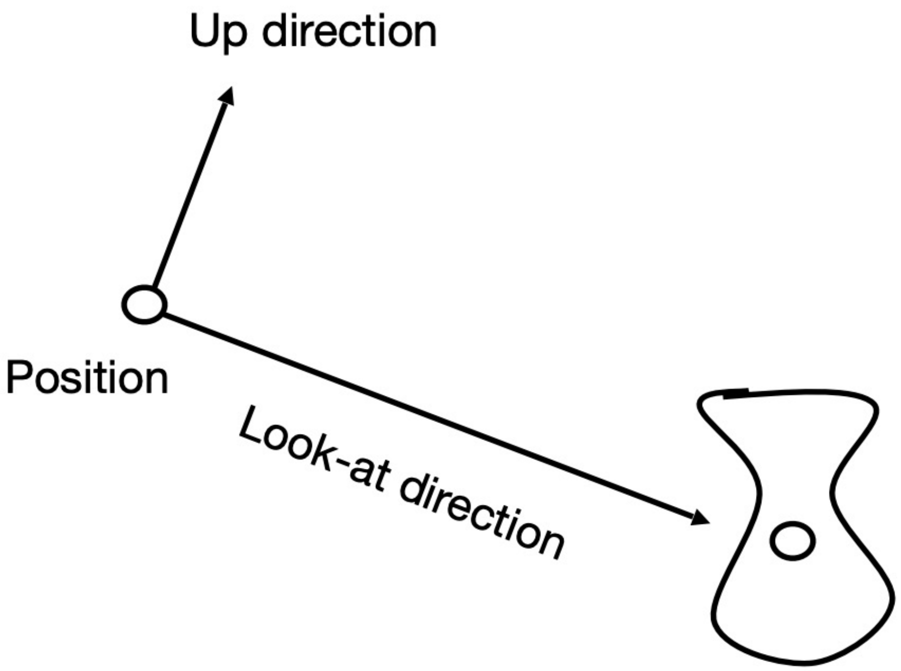
Agenda

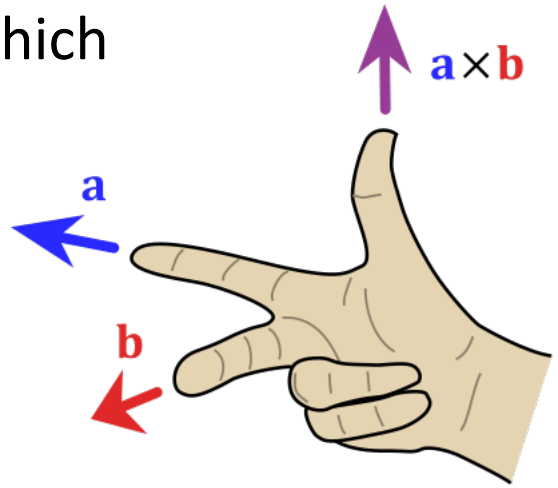
- View Matrix
- Projection Matrix
- About Homework

View Matrix

Camera setting

- Position: $\vec{e}$
- View direction: $\vec{g}$
- Up direction: $\vec{t}$
- Camera coordinate
  - +Z: $-\vec{g}$ (Remember to normalize it)
  - +Y: $\vec{t}$
    - We have usually a reference up direction, which may be not exactly perpendicular to –Z
    - How to compute? Gram-Schmidt
  - +X: $\vec{g} \times \vec{t}$
    - The order matters!





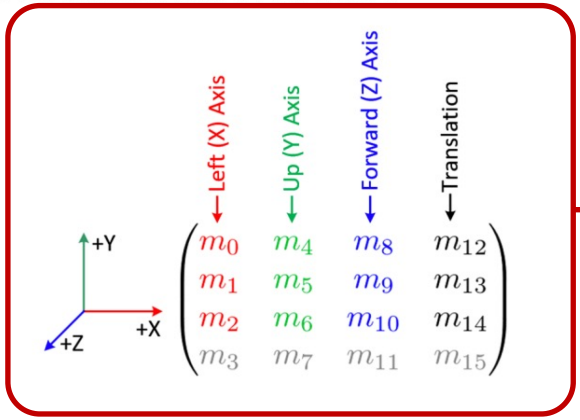
How to find M_{view}？

- How to understand view transformation?
  - In the world coordinate, we have a camera and some objects
  - => We want to represent objects in the view coordinate
  - => We want to know what if we regard the camera coordinate as the new world coordinate
  - => We can simply transform the camera coordinate so that it aligns with the world coordinate
  - The transformation is represented with M_{view}
- How to find M_{view}
  - Two steps: translation and then rotation
  - $M_{view} = R_{view}T_{view}$

This may be a little confusing...

View transformation

- **How to compute the view transform?**
  - Translation + rotation from world coordinate system
  - World coordinate system forms an identity matrix
  - Thus, view matrix is formed by camera coordinate system + camera translation in world coordinates
- **View transformation matrix**



$M_{view} = R_{view} T_{view}$

This does **NOT** mean you can compute M_{view} by set the first column to +X, the second colume to +Y, the third column to +Z, and the last column to the translation.

This mean you can interpret M_{view} as the process that you first rotate the camera according to the first three columns, and then translate it according to the translation column.

How to find T_{view}

- Assume camera is at (x_e, y_e, z_e), we want to translate it to (0, 0, 0)

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

How to find R_{view}

- World coordinate bases: $i = (1,0,0)\,, j = (0,1,0)\,, k = (0,0,1)$
  - p = (x, y, z)
- Camera coordinate bases: i', j', k'
  - p = (x', y', z')

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} i' & j' & k' \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \overbrace{\begin{bmatrix} i' & j' & k' \end{bmatrix}^T}^{R_{view}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- The inverse of an orthonormal matrix is its transpose

$$\begin{bmatrix} i' & j' & k' \end{bmatrix}^{-1} = \begin{bmatrix} i' & j' & k' \end{bmatrix}^T$$

View matrix

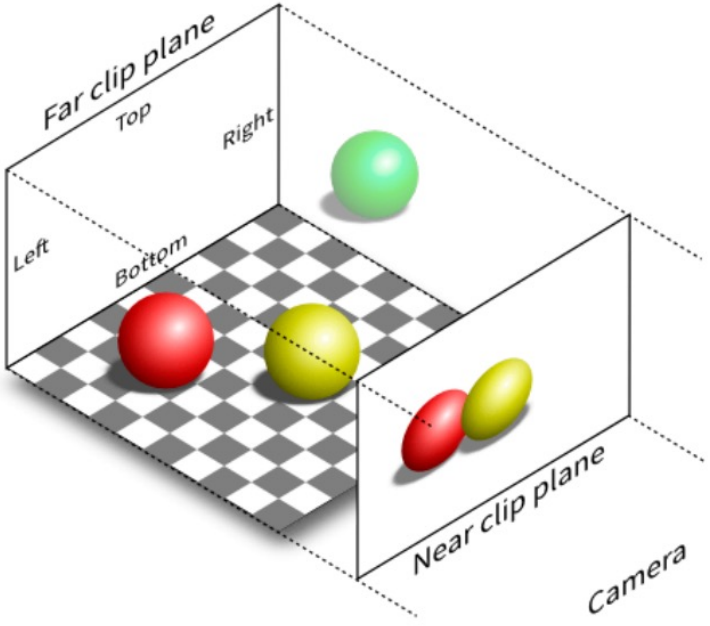
- $M_{view} = R_{view} T_{view}$

$$R_{view} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

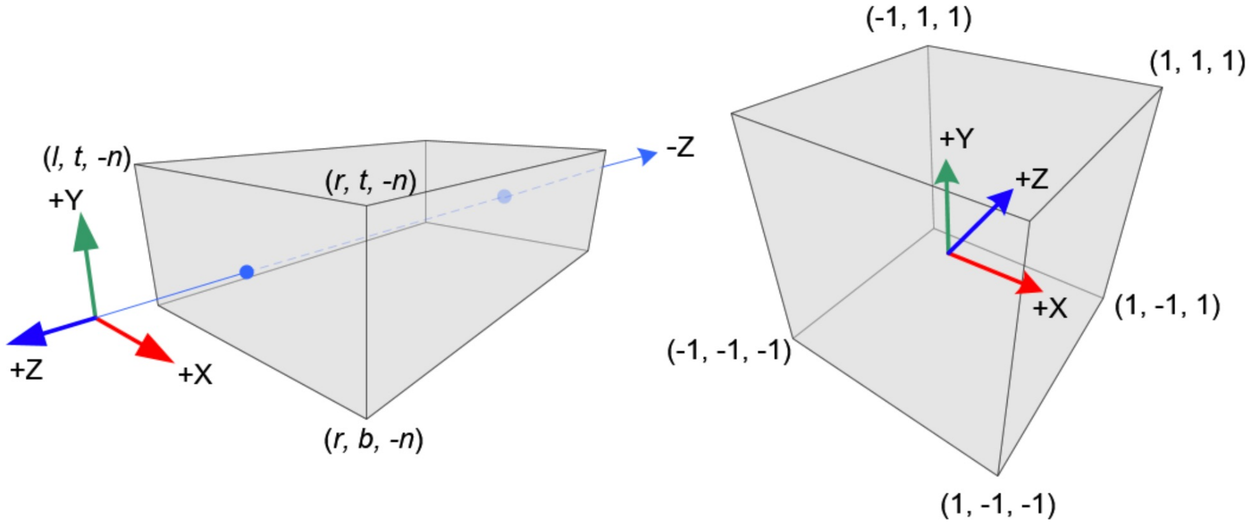
Projection Matix

Orthographic Projection

- What are we going to do?
  - From Eye coordinate to Normalized device coordinate
  - In general, we want to map a rectangular volume to a cube



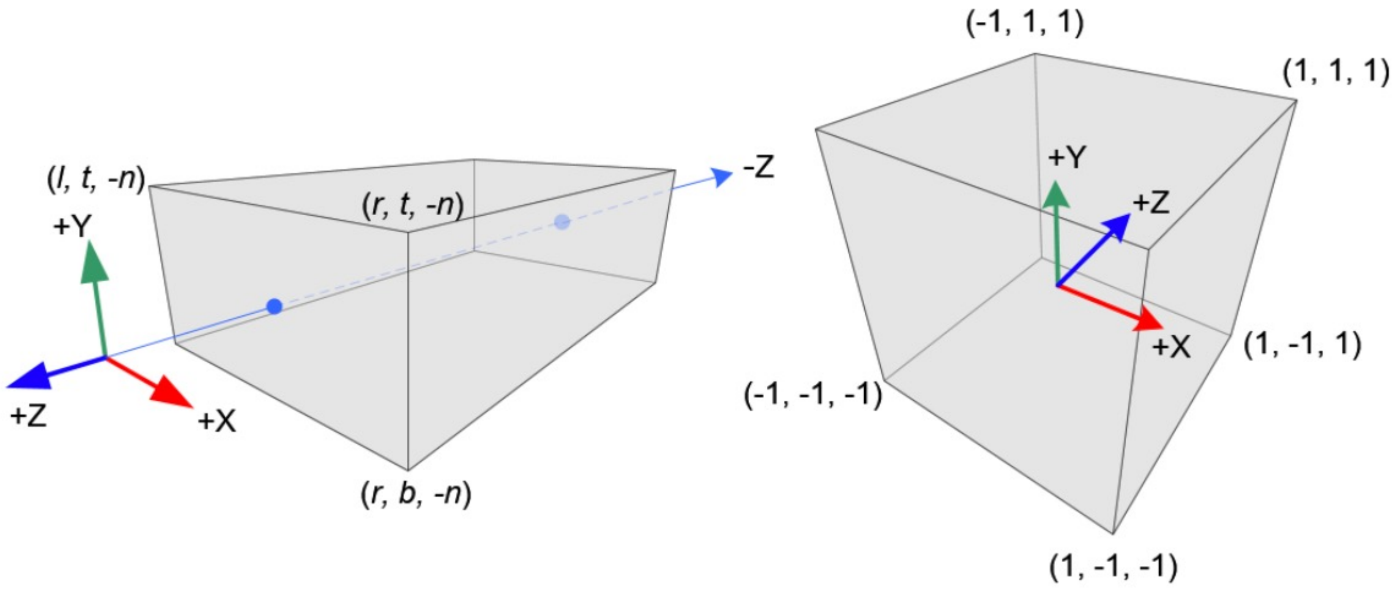
Orthographic projection (O)



Orthographic Volume and Normalized Device Coordinates (NDC)

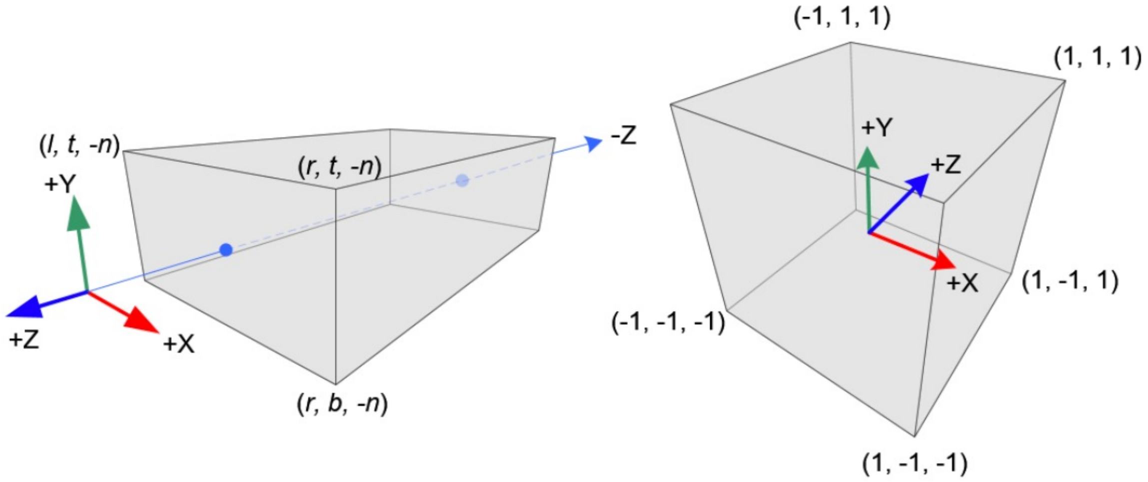
Orthographic Projection

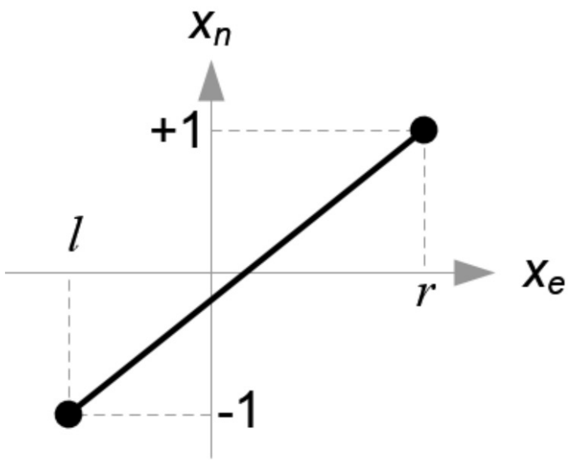
- How to find the projection matrix?
  - Consider the linear relationships of the coordinates below
  - Find the expressions of x_n, y_n, z_n
    - Please notice that eye coordinates are defined in the right-handed coordinate system, but NDC uses the left-handed coordinate system
    - The Z coordinate is inverse



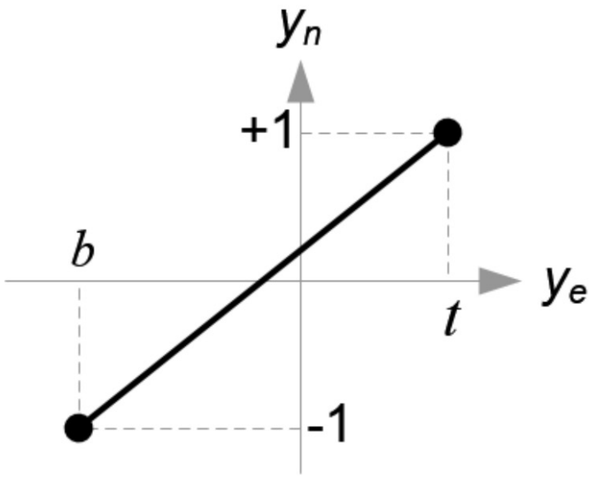
Orthographic Projection

- Find the expressions of x_n, y_n, z_n

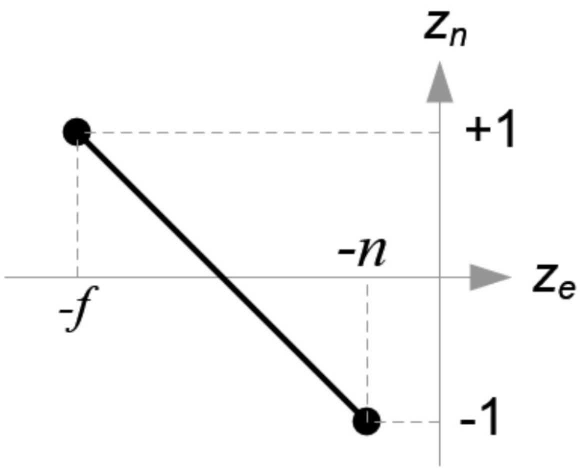




Mapping from x_e to x_n



Mapping from y_e to y_n



Mapping from z_e to z_n

$$x_n = \frac{2}{r-l} \cdot x_e - \frac{r+l}{r-l}$$

$$y_n = \frac{2}{t-b} \cdot y_e - \frac{t+b}{t-b}$$

$$z_n = \frac{-2}{f-n} \cdot z_e - \frac{f+n}{f-n}$$

Orthographic Projection

- How to find the projection matrix?
  - Consider the linear relationships of the coordinates below
  - Find the expressions of x_n, y_n, z_n
  - Use a matrix to represent the linear transform

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}$$

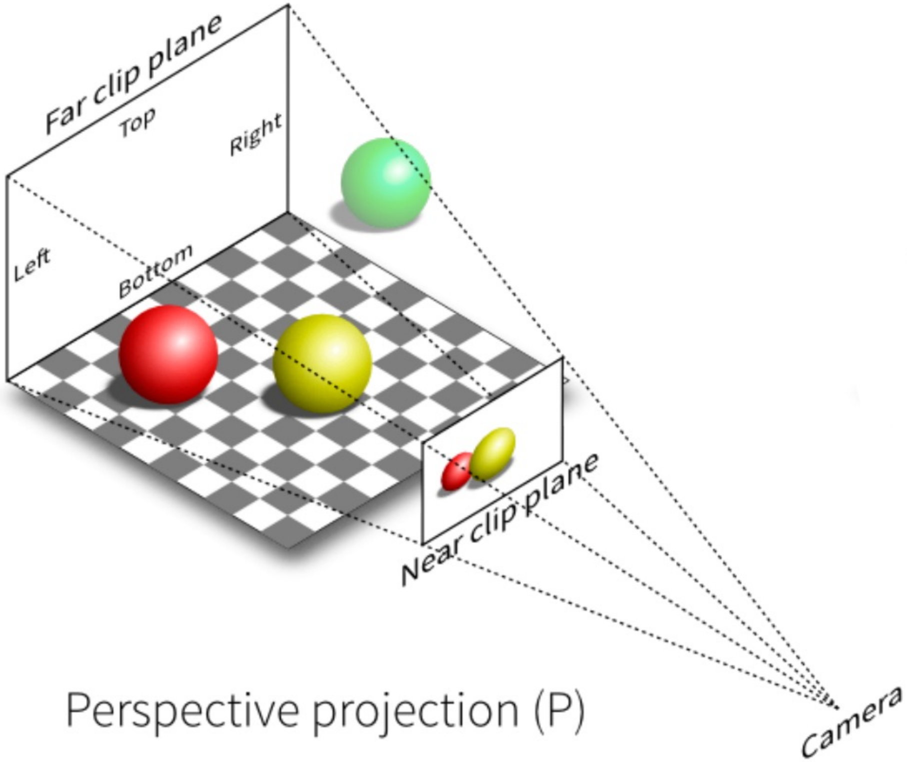
$$x_n = \frac{2}{r-l} \cdot x_e - \frac{r+l}{r-l}$$

$$y_n = \frac{2}{t-b} \cdot y_e - \frac{t+b}{t-b}$$

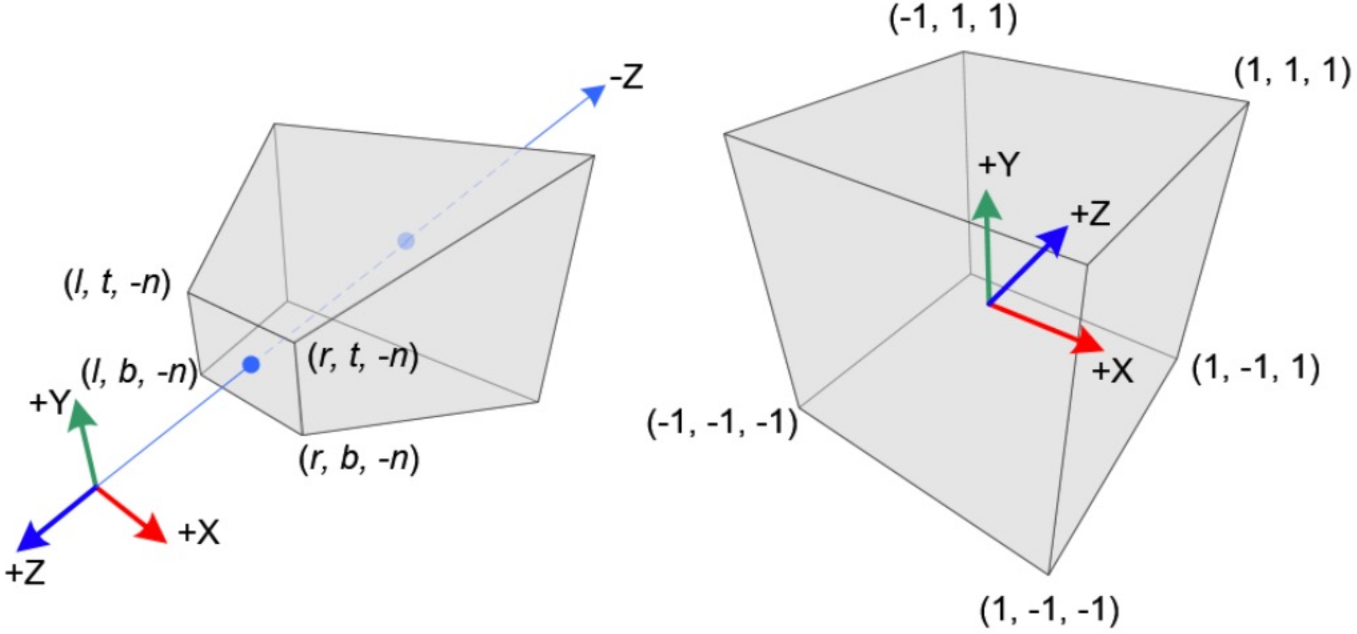
$$z_n = \frac{-2}{f-n} \cdot z_e - \frac{f+n}{f-n}$$

Perspective Projective

- What are we going to do?
  - Mapping a truncated pyramid frustum into a cube.

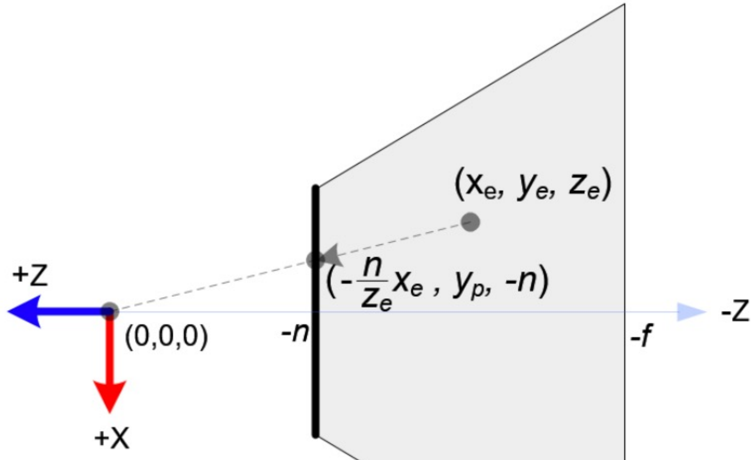


Perspective projection (P)



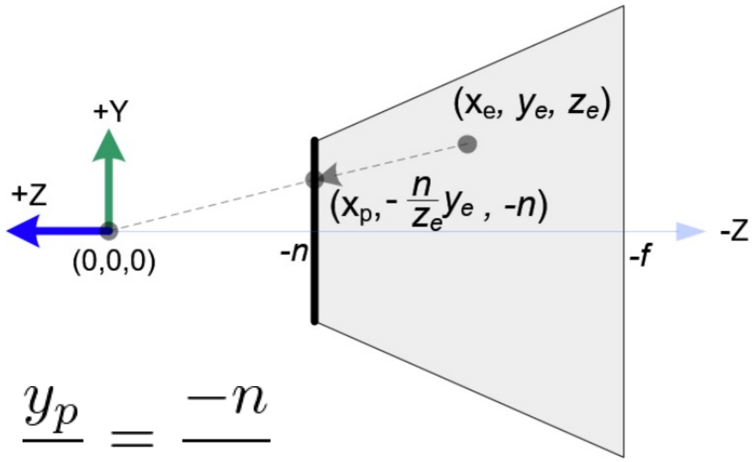
Perspective Projective

- What are we going to do?
  - Mapping a truncated pyramid frustum into a cube.
  - Mapping a point (x_e, y_e, z_e) to the point (x_p, y_p, z_p) in the near plane
  - Both x_p and y_p are inversely proportional to $-z_e$.



$$\frac{x_p}{x_e} = \frac{-n}{z_e}$$

$$x_p = \frac{-n \cdot x_e}{z_e} = \frac{n \cdot x_e}{-z_e}$$



$$\frac{y_p}{y_e} = \frac{-n}{z_e}$$

$$y_p = \frac{-n \cdot y_e}{z_e} = \frac{n \cdot y_e}{-z_e}$$

Perspective Projective

- We have two steps:
  - Eye coordinate to clip coordinate (by Projection matrix)
  - Clip coordinate to normalized device coordinate (by dividing w_{cli})

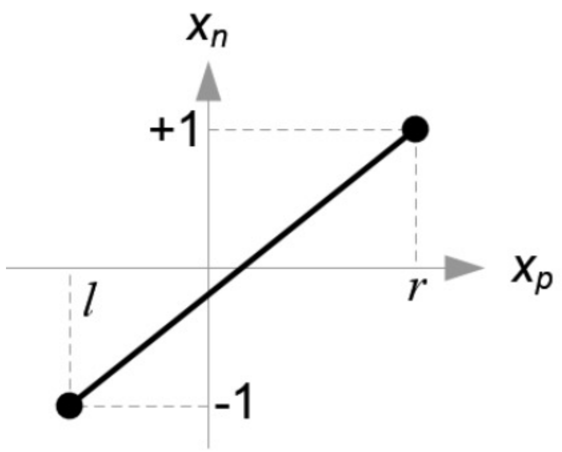
$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = \boxed{M_{projection}} \cdot \begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix} \qquad \begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{pmatrix} = \begin{pmatrix} x_{clip}/w_{clip} \\ y_{clip}/w_{clip} \\ z_{clip}/w_{clip} \end{pmatrix}$$

  - Recall that $x_p = \frac{-n \cdot x_e}{z_e} = \frac{n \cdot x_e}{-z_e} \qquad y_p = \frac{-n \cdot y_e}{z_e} = \frac{n \cdot y_e}{-z_e}$
  - We will constrain w_{clip} to be $-z_e$

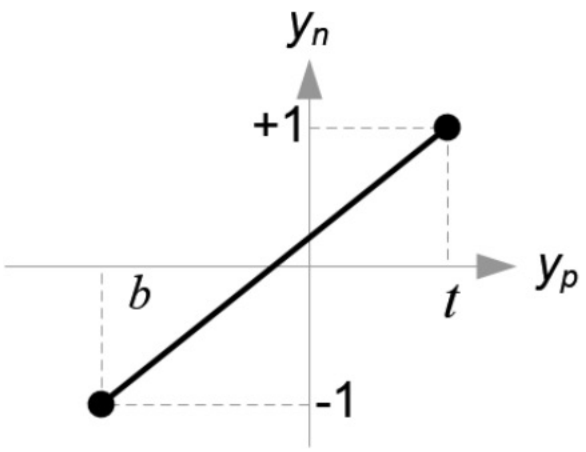
$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}$$

How to find the projection matrix

- We then map x_p and y_p to x_n and y_n of NDC with linear relationship



Mapping from x_p to x_n



Mapping from y_p to y_n

$$x_n = \frac{2x_p}{r-l} - \frac{r+l}{r-l} \qquad \left(x_p = \frac{nx_e}{-z_e}\right)$$
$$= \underbrace{\left(\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e\right)}_{x_c} \Big/ -z_e$$

$$y_n = \frac{2y_p}{t-b} - \frac{t+b}{t-b} \qquad \left(y_p = \frac{ny_e}{-z_e}\right)$$
$$= \underbrace{\left(\frac{2n}{t-b} \cdot y_e + \frac{t+b}{t-b} \cdot z_e\right)}_{y_c} \Big/ -z_e$$

How to find the projection matrix

- Then we have the matrix form

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}$$

$$x_n = \frac{2x_p}{r-l} - \frac{r+l}{r-l} \qquad (x_p = \frac{nx_e}{-z_e})$$

$$= \Bigg(\underbrace{\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e}_{x_c} \Bigg) \Big/ - z_e$$

$$y_n = \frac{2y_p}{t-b} - \frac{t+b}{t-b} \qquad (y_p = \frac{ny_e}{-z_e})$$

$$= \Bigg(\underbrace{\frac{2n}{t-b} \cdot y_e + \frac{t+b}{t-b} \cdot z_e}_{y_c} \Bigg) \Big/ - z_e$$

How to find the projection matrix

- To find the 2 rest elements, we need 2 equations

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}$$

$$z_n = z_c / w_c = \frac{A z_e + B w_e}{-z_e}$$

- Near plane: $(x_e, y_e, -n, 1) \to z_n = -1$
- Far plane: $(x_e, y_e, -f, 1) \to z_n = 1$

$$\begin{cases} \dfrac{-An + B}{n} = -1 \\ \dfrac{-Af + B}{f} = 1 \end{cases}$$

How to find the projection matrix

- Congratulations!!!

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

About Homework

Warm-up Assignment

- Do NOT forget to finish your report.
- Describe your implementations and show the result in a more detailed way.

Assignment 1

DDL: 22:00, Oct 7, 2021

Programming Requirements

- **[must]** You are required to load mesh objects from files and draw the meshes. (40%)
- **[must]** You are required to render objects with a Phong lighting model. (30%)
- **[must]** You are required to manipulate the camera and use the keyboard to control the camera: you can use the keyboard to translate and rotate the camera so that you can walk in the virtual scene. (30%)
- **[optional]** You can accomplish the above rendering requirements by utilizing the modern OpenGL with shaders.
- **[optional]** You can use a fragment shader to support multiple lights.
- **[optional]** You can use a geometry shader to change the vertex data.

Data arrangement in .object file

vertex count	normal count	face count
int	int	int

vertex count

float float float
…
float float float

position of a vertex

normal count

float float float
…
float float float

normal of a vertex

face count

int int int int int int
…
int int int int int int

The position and normal indices of vertices in one face

How to read from files

- ifstream

```
void loadDataFromFile(const std::string &path) {
  std::ifstream fin;
  fin.open(path);

  if (fin.is_open()) {
    int vertex_count, normal_count, face_count;
    fin >> vertex_count >> normal_count >> face_count;
    std::cout << vertex_count << " " << normal_count << " " << face_count << std::endl;
  }

  fin.close();
}
```

Camera navigation

- Use some functions to capture your keyboard events
  - glfwGetKey(…)
- You may also need to define some callback functions
  - mouse_callback(GLFWwindow* window, double xpos, double ypos)
  - scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
- Update the view matrix
  - To construct the view matrix in glm: glm::LookAt(…)
  - If you do not use a vertex shader, consider gluLookAt(…)
- For more detailed information:
  - https://learnopengl.com/Getting-started/Camera

Good Luck!