Laborator 9

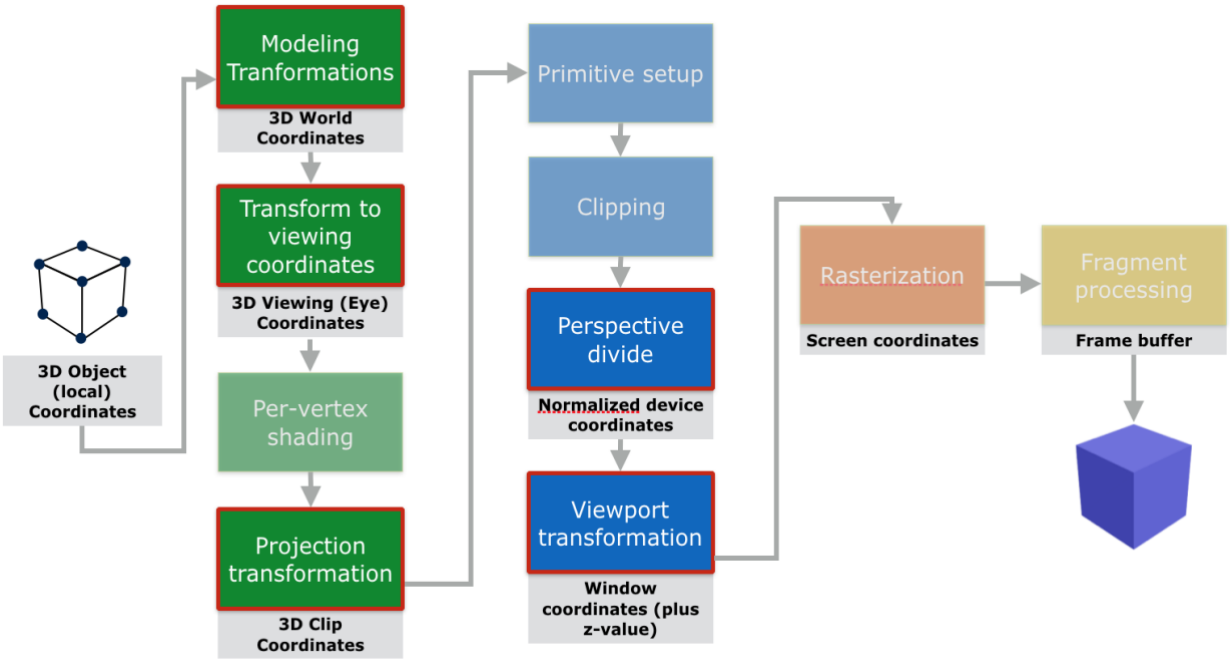
1 Obiective

Acest laborator prezintă transformările de vizualizare utilizate pentru a mapa locațiile 3D (specificate prin coordonatele x, y și z) la coordonatele 2D (specificate prin coordonatele pixelilor) ale ecranului.

2 Transformări de vizualizare

Pentru a mapa coordonatele 3D la coordonate 2D, folosim o secvență de trei transformări:

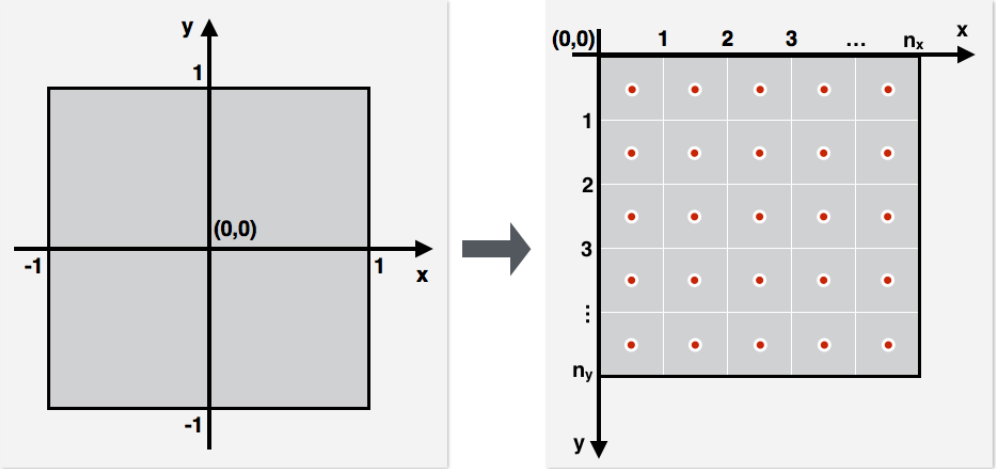
- **Transformări ale camerei**
  - Folosite pentru a plasa camera în origine și pentru a repoziționa toate celelalte obiecte în raport cu noua poziție a camerei
  - Această transformare depinde numai de poziția și orientarea camerei
- **Transformări de proiecție**
  - Utilizat pentru a proiecta puncte din spațiul de coordinate ale camerei
  - După transformare toate punctele vizibile vor fi în intervalul [-1, 1]
  - Această transformare depinde numai de tipul de proiecție (perspectivă sau ortogonală)
- **Transformări de viewport**
  - Folosite pentru a mapa dreptunghiul unitate din spațiul imaginii la dreptunghiul dorit în coordonate pixeli
  - Această transformare depinde numai de mărimea și poziția imaginii rezultate



Pe parcursul acestor transformări schimbăm sistemele de coordonate în care sunt specificate obiectele. Transformarea camerei schimbă coordonatele din **spațiul global** (world space) în **spațiul camerei**. Transformarea de proiecție mută puncte din **spațiul camerei** în **volumul de vizualizare canonic** (aici decuparea este efectuată mai eficient). Ultima transformare, transformarea ferestrei de vizualizare (viewport) mapează **volumul de vizualizare canonic** la **spațiul ecran** (screen space).

3 Transformarea ferestrei de vizualizare (Viewport)

Această transformare este utilizată pentru a mapa puncte din volumul de vizualizare canonic (unde valorile sunt în intervalul [-1, 1]) la spațiul ecran (definit de lățimea și înălțimea imaginii rezultate). Aceasta se compune din mai multe transformări, inclusiv translație, scalare și reflecție. Originea imaginii este considerată colțul din stânga sus. Pentru alte specificații ale spațiului imaginii, transformările ar putea fi diferite. Centrul pixelilor este considerat a fi la valoarea întregă plus 0,5 (ambele pe axa x și y).



$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & 0 \\ 0 & \frac{n_y}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

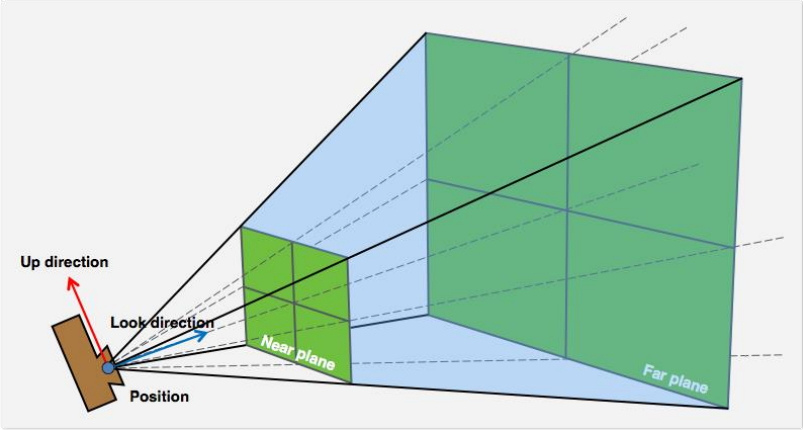
$$\mathbf{M}_{vp} = \mathbf{SMT} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x}{2} \\ 0 & -\frac{n_y}{2} & 0 & \frac{n_y}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pentru a afișa imaginea pornind de la o altă locație decât punctul (0, 0) trebuie să adăugăm o transformare de translație adițională matricii $\mathbf{M}_{vp}$.

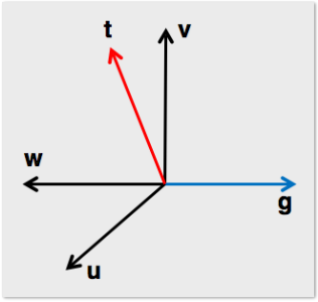
M_{vp} = T(startx, starty)*S*M*T

4 Transformarea camerei

O cameră virtuală este definită de un set de atribute și parametri, cum ar fi: Poziția camerei, Orientare, Câmpul de vizualizare, Tipul de proiecție (perspectivă sau paralelă), Adâncimea câmpului, Distanța focală, Înclinarea și decalarea lentilei camerei relative la corpul camerei. Pentru un model de bază al camerei trebuie să specificăm cel puțin poziția acesteia, orientarea și câmpul vizual.



Camera este specificată în coordonate globale prin următorii parametri: poziția **e**, direcția de vizitare **g**, vectorul de vizualizare **t**. Din acești parametri trebuie să definim un sistem de coordonate **uvw**.



$$\mathbf{w} = -\frac{\mathbf{g}}{\|\mathbf{g}\|}$$

$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|}$$

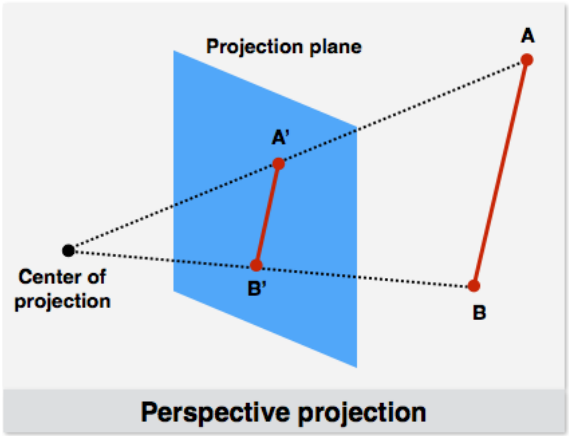
$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

Trebuie să aliniem cele două sisteme de coordonate diferite (axele u, v, w cu axele x, y, z) folosind următoarea matrice de transformare:

$$\mathbf{M}_{cam} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

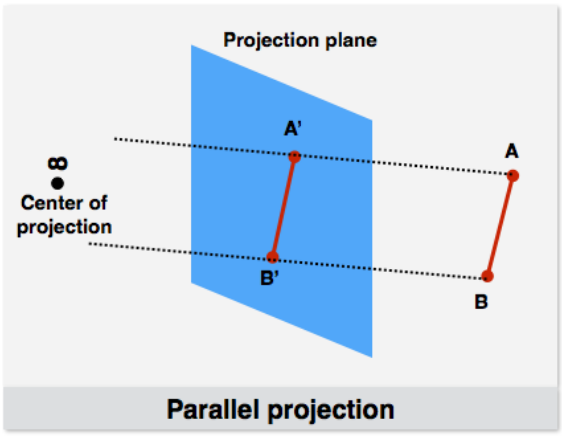
5 Transformarea de proiecție

Două tipuri de proiecții sunt prezentate în acest document, proiecția perspectivă și cea paralelă.



Proiecția perspectivă:

- Razele de proiecție converg în centrul proiecției (locația observatorului)
- Obiectele apar mai mici cu creșterea distanței față de centrul proiecției (ochiul observatorului)
- Liniile paralele cu planul de proiecție rămân paralele
- Liniile care nu sunt paralele cu planul de proiecție converg către un singur punct (punct de dispariție)



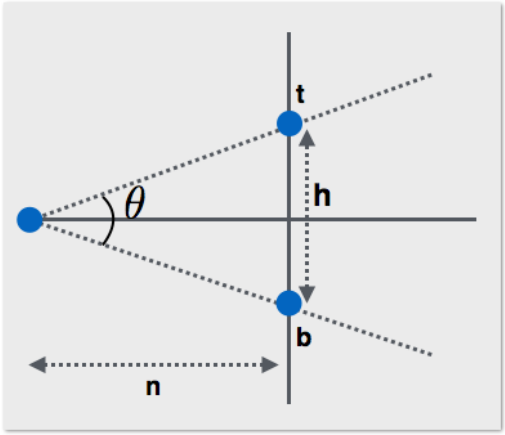
Proiecția paralelă:

- Raze de proiecție paralele
- Punctul de convergență la infinit
- Poziția observatorului la infinit
- Liniile paralele rămân paralele

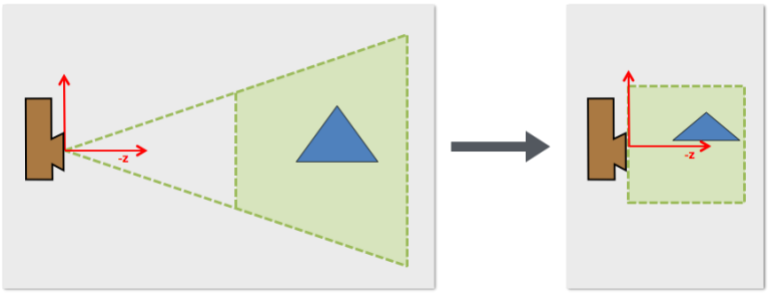
$$\mathbf{M}_{per} = \begin{bmatrix} -\frac{1}{aspect \tan\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & -\frac{1}{\tan\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\tan\frac{\theta}{2} = \frac{h}{2|n|}$$

$$aspect = \frac{width}{height} = \frac{r-l}{t-b}$$



Înainte de a afișa nodurile, trebuie să efectuăm o operație denumită **diviziune de perspectivă**.



Planele de apropiere și de depărtare sunt definite în coordonatele camerei. Deoarece camera este localizată în origine și este orientată în direcția negativă z, valorile de apropiere și de depărtare ar trebui să fie negative, cu n(near) > f (far).

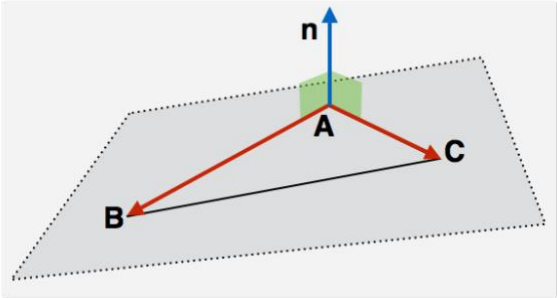
$$\mathbf{P} = \mathbf{M}_p \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} xd \\ yd \\ zd \\ z \end{bmatrix} \xrightarrow{\text{After perspective divide}} \begin{bmatrix} \frac{d}{z}x \\ \frac{d}{z}y \\ d \\ 1 \end{bmatrix}$$

6 Combinarea transformărilor

```
construim Mvp
construim Mper
construim Mcam
M = Mvp * Mper * Mcam
for each segment de linie(ai,bi) do
   p = M * ai
   q = M * bi
   drawline(xp/wp,yp/wp,xq/wq,yq/wq)
```

7 Calculul și afișarea vectorului normală al unui triunghi

Pentru a calcula vectorul normală, putem folosi următoarea formulă, bazată pe produsul vectorial dintre două vectori. Ultimul pas este de a normaliza vectorul rezultat pentru a obține un vector normală.

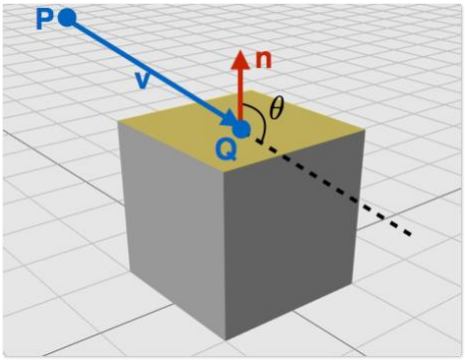


$$\mathbf{n} = (B - A) \times (C - A)$$

Pseudocodul pentru afișarea vectorului normală:

```
displayNormalVector()
   centerPoint = calculati punctul central al triunghiului
   normalVector = calculati vectorul normala
   secondPoint = centerPoint + normalVector * offset
   drawLine(centerPoint, secondPoint)
```

8 Eliminarea fețelor orientate invers



În funcție de poziția relativă a camerei față de triunghiurile obiectului, putem identifica triunghiurile vizibile.

Poziția relativă a unui punct **P** (poziția camerei) față de un plan (triunghi):

- θ > 90˚ atunci P se află în fața planului
- θ = 90˚ atunci P se află pe plan
- θ < 90˚ atunci P se află în spatele planului

Unghiul poate fi calculat folosind produsul scalar dintre cei doi vectori, **v** și **n**.

9 Decuparea în coordonate omogene

Putem decupa punctele pe baza următoarelor plane de decupare:

-P.w <= P.x <= P.w
-P.w <= P.y <= P.w
-P.w <= P.z <= P.w

10 Temă

Descărcați codul sursă și implementați următoarele metode (din projection.h):

- bool clipPointInHomogeneousCoordinate(const egc::vec4 &vertex)
- bool clipTriangleInHomegeneousCoordinates(const std::vector<egc::vec4> &triangle)
- egc::vec3 findNormalVectorToTriangle(const std::vector<egc::vec4> &triangle)
- egc::vec4 findCenterPointOfTriangle(const std::vector<egc::vec4> &triangle)
- bool isTriangleVisible(const std::vector<egc::vec4> &triangle, const egc::vec3 &normalVector)
- void displayNormalVectors(egc::vec3 &normalVector, egc::vec4 &triangleCenter)
- mat4 defineViewTransformMatrix(int startX, int startY, int width, int height)
- mat4 defineCameraMatrix(Camera myCamera)
- mat4 definePerspectiveProjectionMatrix(float fov, float aspect, float zNear, float zFar)
- void perspectiveDivide(vec4 &inputVertex)