Facultatea de
Matematică și Informatică
Universitatea din București

Programarea Aplicațiilor de Simulare

Transformări și Randare

Curs 7

Pătrânjel David-George

Agenda Cursului

01

Geometrie

Poligoane

Transformarea Geometriei

02

Randare

Reprezentarea culorilor

Rasterizare

03

Transformări

Spațiul Obiectului

Spațiul Lumii

Spațiul Perspectivă

04

OpenGL Pipeline

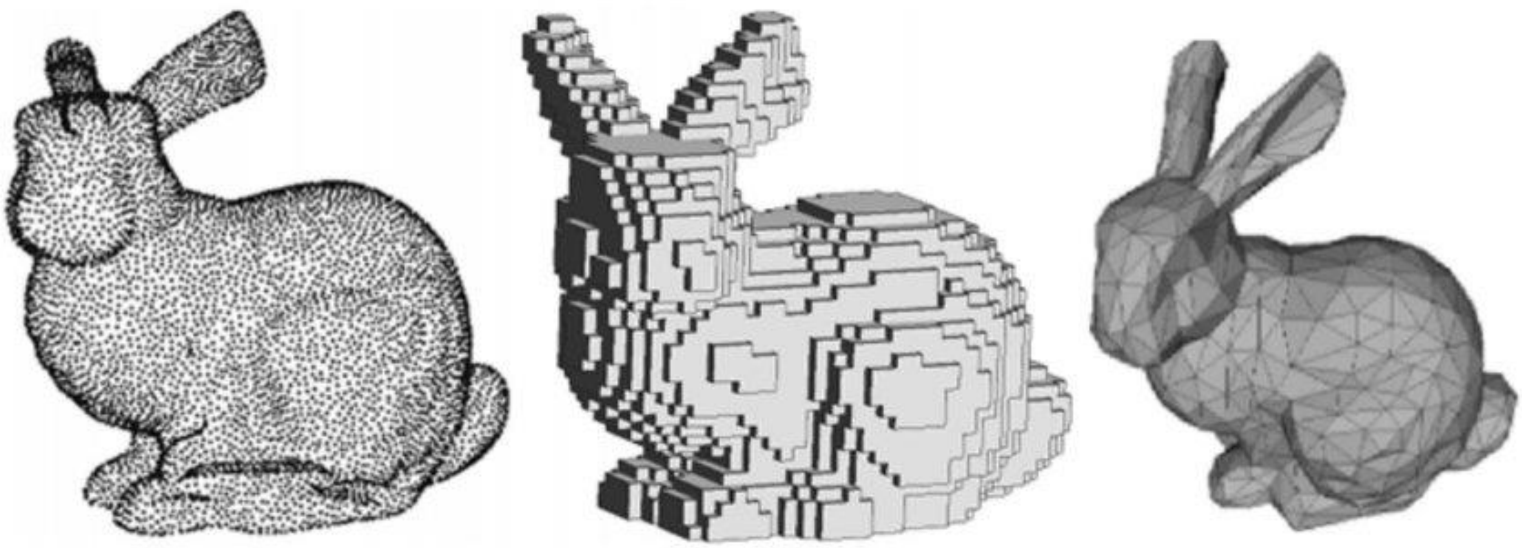
01

Geometrie

Poligoane

Transformarea Geometriei

Cum putem descrie suprafețe pentru desenarea grafică?



Reprezentarea unui obiect 3D

Lahoud, Jean & Cao, Jiale & Khan, Fahad & Cholakkal, Hisham & Anwer, Rao & Khan, Salman & Yang, Ming-Hsuan. (2022). 3D Vision with Transformers: A Survey. 10.48550/arXiv.2208.04309.

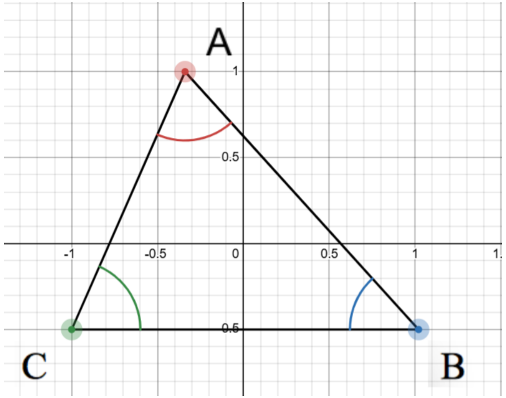
https://en.m.wikipedia.org/wiki/File:Stanford_bunny_qem.png

Poligoane

Putem defini obiecte în 2D și în 3D prin definirea poligoanelor din care sunt alcătuite.

Triunghiuri

- În grafică (și jocuri), poligoanele sunt reprezentate de triunghiuri deoarece placa video are implementat în hardware desenarea acestora.
- Un triunghi este alcătuit din **3 vârfuri** (en. vertices).
- Un vârf poate conține mai multe informații: poziția, culoarea, normala etc.
- Pentru topologii complexe se folosesc **două liste: vârfuri și indici.**
- Se încărca o singură dată în memoria plăcii video toate vârfurile geometriei, iar apoi putem specifica poligoanele pe care acestea le formează folosind indici.



Triunghiuri

```
float vertices[] = {
        0.5f,  0.5f, 0.0f,  // top right
        0.5f, -0.5f, 0.0f,  // bottom right
        -0.5f, -0.5f, 0.0f,  // bottom left
        -0.5f,  0.5f, 0.0f   // top left };
unsigned int indices[] = {
        0, 1, 3,  // first triangle
        1, 2, 3   // second triangle};
```

Declararea listelor de vârfuri și indecși

Transmiterea datelor spre GPU

```
unsigned int vbo ;
glGenBuffers (1 , vbo);
glBindBuffer(GL_ARRAY_BUFFER, vbo);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices , GL_STATIC_DRAW);

unsigned int ebo;
glGenBuffers(1 , ebo );
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices , GL_STATIC_DRAW);
```

02

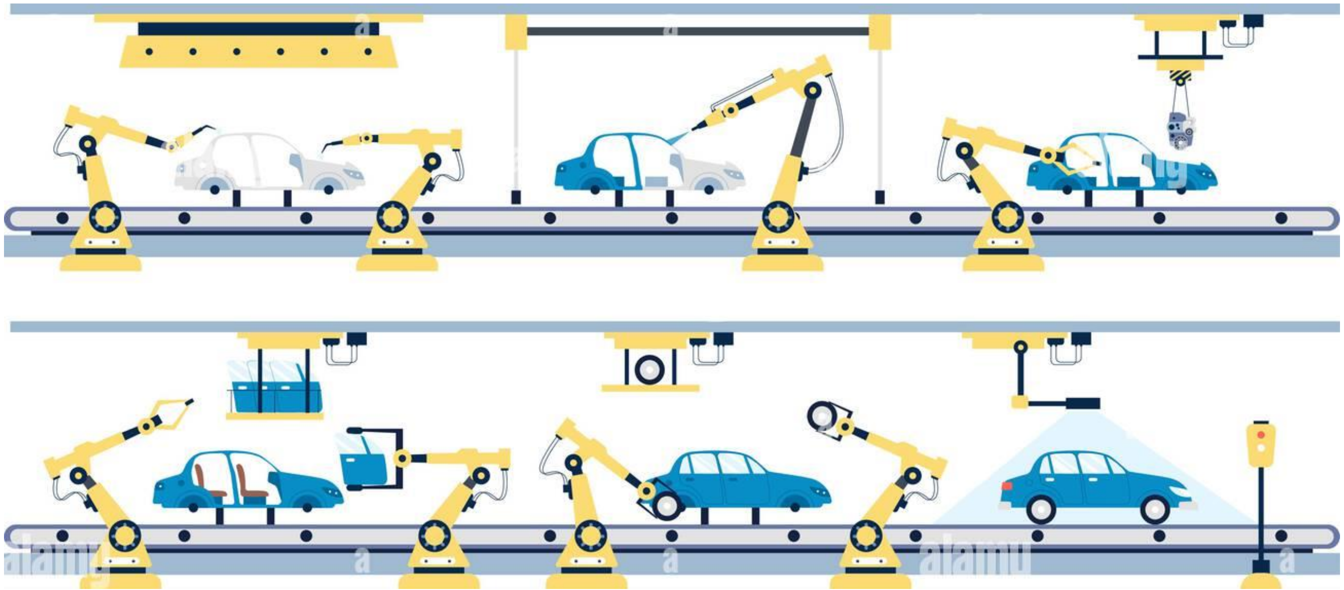
Randare

Reprezentarea culorilor

Rasterizare

Randare

Randarea/Banda grafică (en. Rendering Pipeline) reprezintă secvența de pași ce sunt realizați pentru crearea imaginii unui cadru.



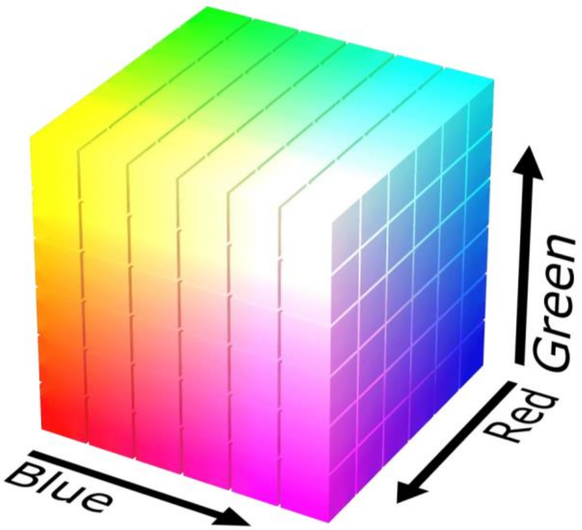
Spațiul culorilor

Culorile sunt obținute combinând internsitățile de pe 3 canale : **R (red); G (green); B (blue) Cubul RGB**

Acestea au de regulă valori normalizate (între 0 și 1).

Exemple:

- (0,1,0) reprezintă culoarea verde.
- (1,1,0) reprezintă culoarea galben.

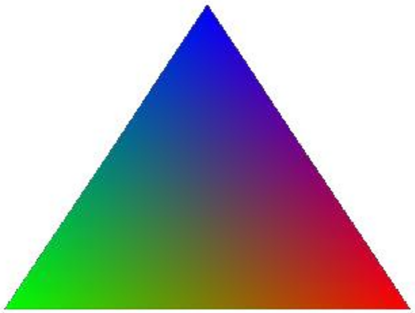


Vârfuri

- O informație pe care o putem atașa vârfurilor este **culoarea acestora**.

- Astfel, putem avea vârfuri care să specifice pozițiile vârfurilor și culorile acestora.

```
float vertices[] =
{
    // positions          // colors
    0.5f, 0.5f, 0.0f,     1.0f, 0.0f, 0.0f,     // bottom right
   -0.5f, 0.5f, 0.0f,     0.0f, 1.0f, 0.0f,     // bottom left
    0.0f, 0.5f, 0.0f,     0.0f, 0.0f, 1.0f      // top
};
```



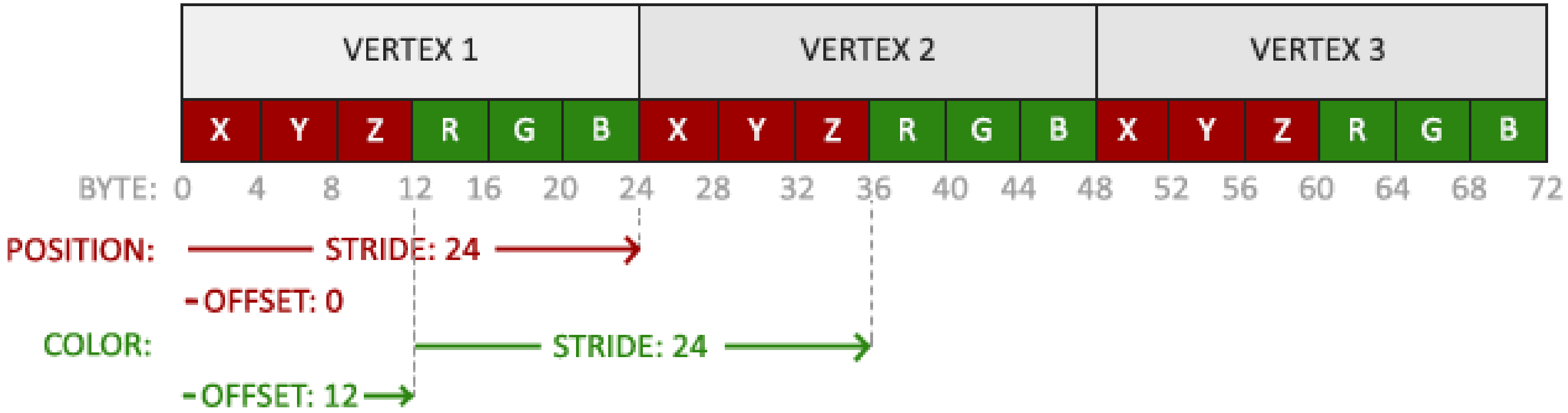
Vârfuri

```
unsigned int vbo;
glGenBuffers(1, &vbo);
glBindBuffer(GL_ARRAY_BUFFER, vbo);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);

// Position attribute
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);

// Color attribute
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)(3 * sizeof(float)));
glEnableVertexAttribArray(1);
```

Vârfuri



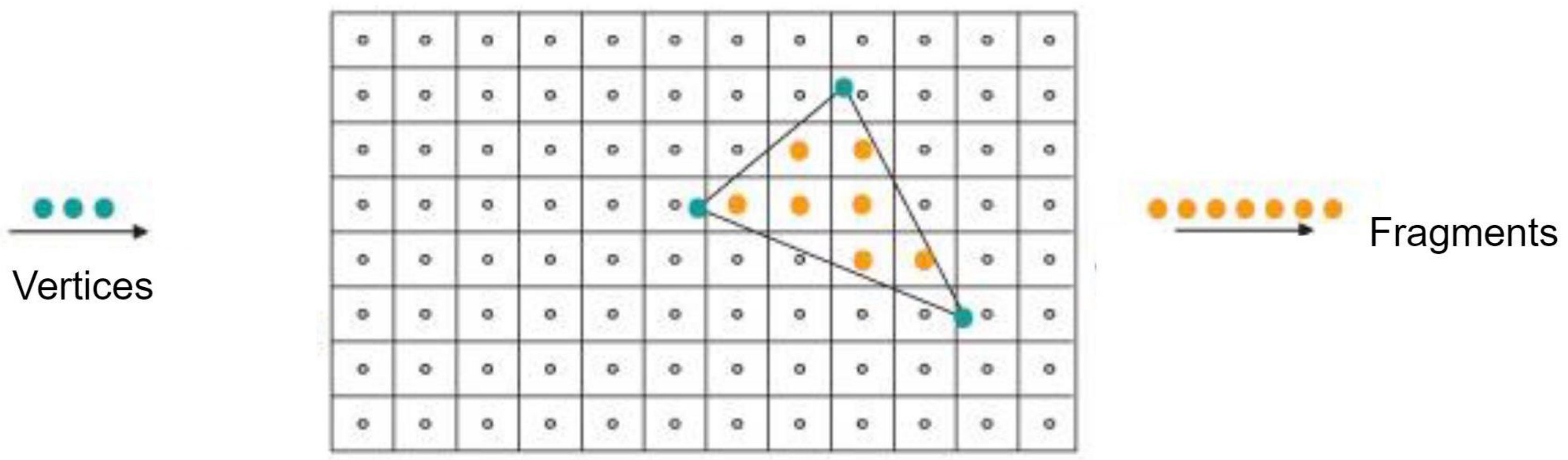
VERTEX 1
VERTEX 2
VERTEX 3
X Y Z R G B X Y Z R G B X Y Z R G B
BYTE: 0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72
POSITION:
STRIDE: 24
OFFSET: 0
COLOR:
STRIDE: 24
OFFSET: 12

Rasterizare

Rasterizarea reprezintă pasul de desenare în grila de pixeli a ecranului a unor primitive (triunghiuri) în spațiul 2D.

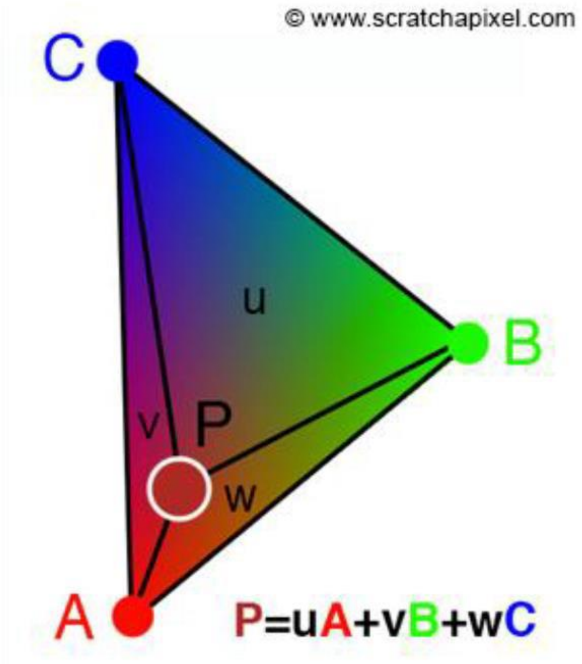
Rasterizare

- Se determină pixelii din interiorul triunghiului
- Proprietățile vertecșilor sunt interpolate liniar.



Interpolarea folosind coordonate baricentrice

- **Combinație liniară a proprietăților vertecșilor**
- Ex: culoare, coordonate de textură, normală, tangente.
- Ponderile sunt direct proporționale cu ariile triunghiurilor determinate de punctul P și laturile opuse ale triunghiului principal.



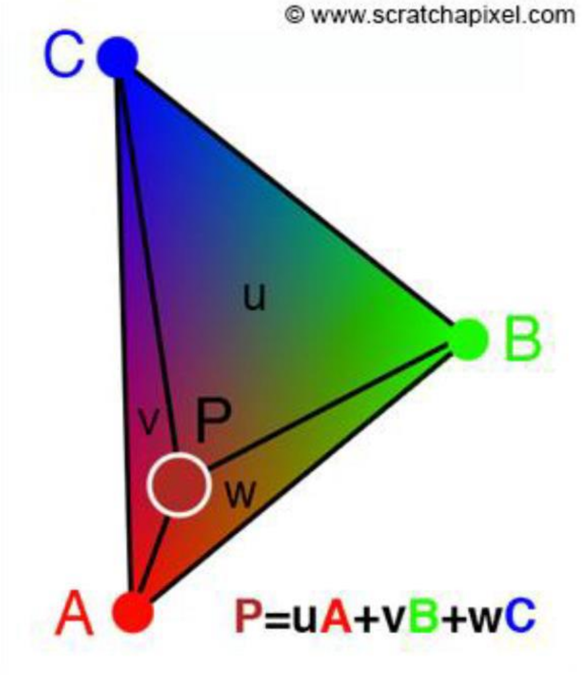
Interpolarea folosind coordonate baricentrice

$$P' = (u, v, w)$$

$$u = \frac{A_{\Delta P V_1 V_2}}{A_{\Delta V_1 V_2 V_3}}$$

$$v = \frac{A_{\Delta P V_1 V_3}}{A_{\Delta V_1 V_2 V_3}}$$

$$w = \frac{A_{\Delta P V_2 V_3}}{A_{\Delta V_1 V_2 V_3}}$$



Anticrenelare (Anti-aliasing)

Efectul de **aliasing** apare din cauza subeșantionării imaginii, rezultând desenarea imaginii cu margini zimțate, linii neregulate sau artefacte vizuale nedorite.

Procesul de anticrenelare (anti-aliasing) este o tehnică utilizată în grafică pentru a reduce efectul de aliasing.

Metode de anti-aliasing:

- Supersampling Anti-Aliasing (SSAA)
- Multisampling Anti-Aliasing (MSAA)
- Fast Approximate Anti-Aliasing (FXAA)

https://www.geeksforgeeks.org/antialiasing/

Anticrenelare prin SSAA

https://vr.arvilab.com/blog/anti-aliasing

Anticrenelare prin MSAA

https://www.youtube.com/watch?v=aCkUHBw7KIg

Anticrenelare prin MSAA

https://www.youtube.com/watch?v=krGX9NdNyMY

03

Transformări

Spațiul Obiectului
Spațiul Lumii
Spațiul Perspectivă

Transformări

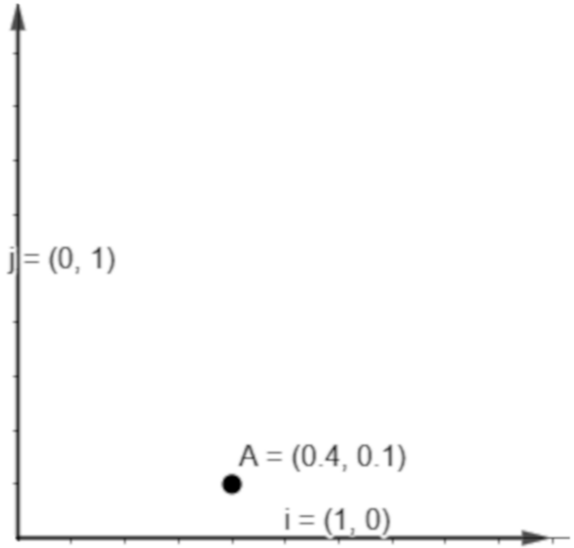
- Sunt operații fundamentale în sinteza imaginilor.
- Folosite pentru:
    - Redarea desenelor/obiectelor 2D sau 3D la diferite mărimi;
    - Compunerea desenelor/scenelor 3D din mai multe obiecte;
    - Realizarea animației;
    - Transformarea obiectelor dintr-un sistem de coordinate în alt sistem de coordonate;
    - - Etc.

Coordonate Carteziene

- Putem considera puncte de forma $\mathbf{P}=(\boldsymbol{x},\boldsymbol{y})$.
- $\vec{i} = \mathbf{1,0}$
- $\vec{j}=[\mathbf{0,1}]$
- $\boldsymbol{P}=\boldsymbol{x}\vec{i}+\boldsymbol{y}\vec{j}$

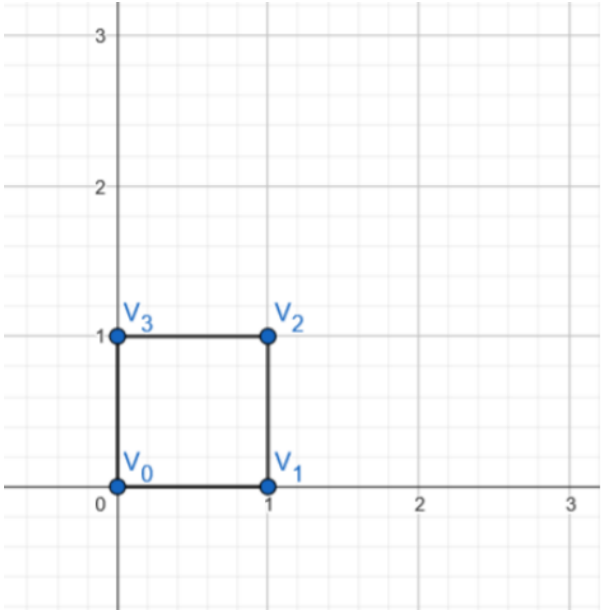
Exemplu:

- fie A=(0.4,0.1)

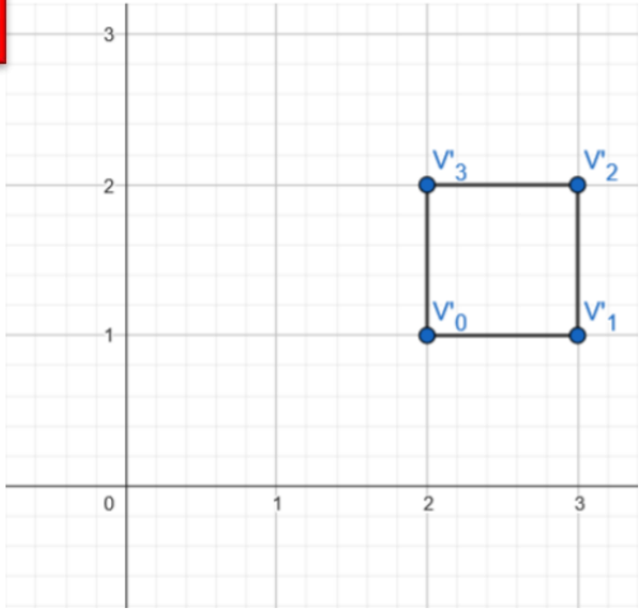


Translație

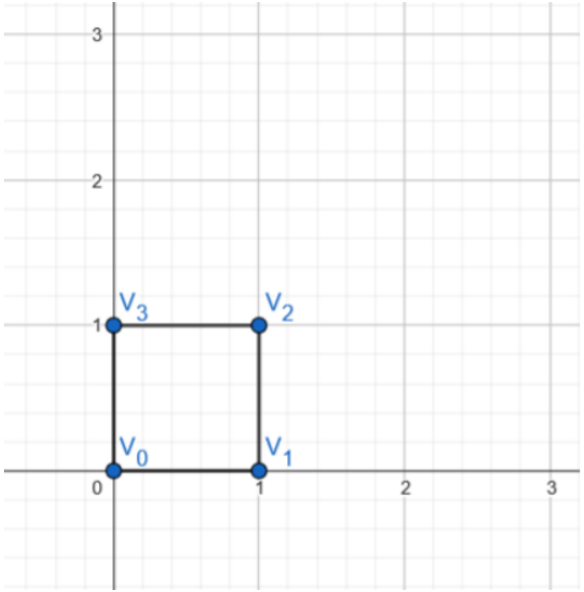
$$x' = x + t_x$$
$$y' = y + t_y$$



translație cu
$\vec{t}$ = [2 1]

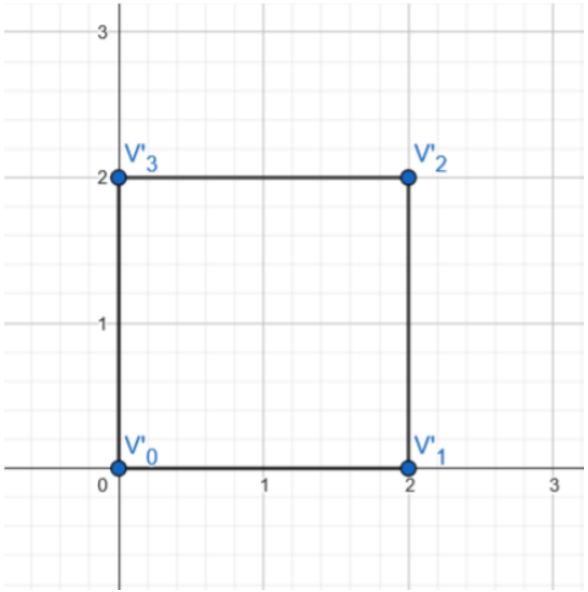


Modificarea Scării



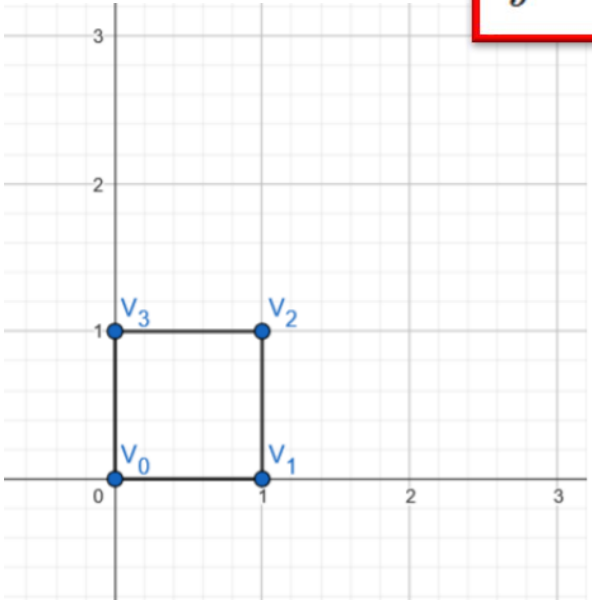
$$\boxed{\begin{aligned} x' &= s_x \cdot x \\ y' &= s_y \cdot y \end{aligned}}$$

modificare scară cu $\vec{s}$ = [2 2]

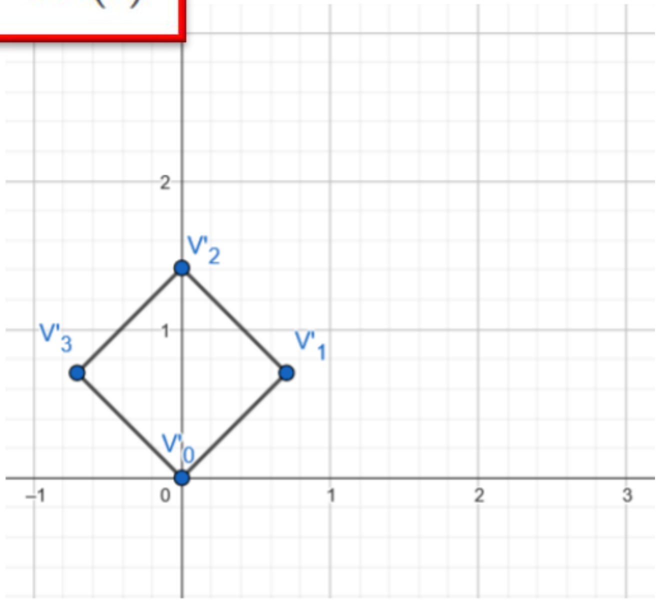


Rotație

$$x' = x \cdot cos(\theta) - y \cdot sin(\theta)$$
$$y' = x \cdot sin(\theta) + y \cdot cos(\theta)$$



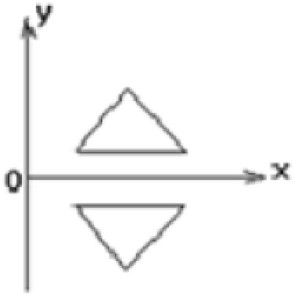
rotație cu
θ=45°



Simetria/Oglindirea

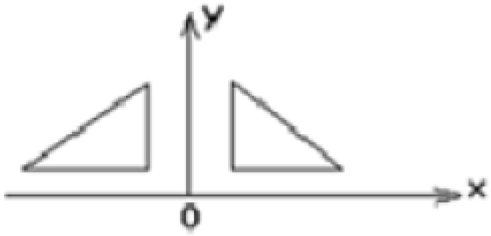
față de OX

$$x' = x$$
$$y' = -y$$



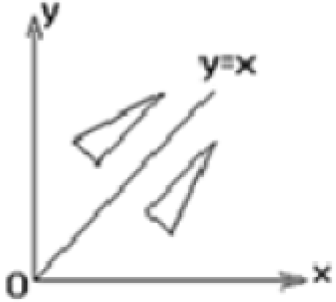
față de OY

$$x' = -x$$
$$y' = y$$



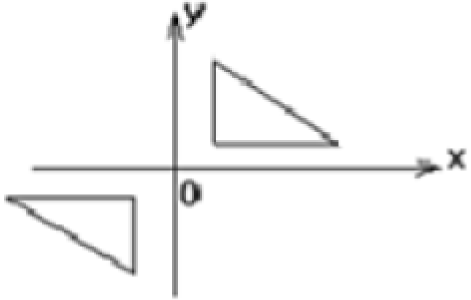
față de dreapta y=x

$$x' = y$$
$$y' = x$$



față de Origine

$$x' = -x$$
$$y' = -y$$



Transformări

- Dorim o reprezentare matricială a transformarilor, pentru compunerea sa cu alte transformari.
- Un punct din plan, P(x, y), se reprezintă în coordonate carteziene printr un vector [x, y] sau $\begin{bmatrix} x \\ y \end{bmatrix}$
- **Transformări liniare:**
  - înmulțire de matrici de forma P’ = M*P;
- **Transformări afine:**
  - înmulțire de matrici de forma P’ = M*P + T;

- Ce facem în cazul translației?

Putem folosi coordonate omogene pentru a reprezenta transformările 2D matricial (3x3)

Matricea de translație

$$[x'\,y'\,1] = [x\,y\,1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \quad \text{sau} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Matricea de modificare a scării

$$[x'\,y'\,1] = [x\,y\,1]\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{sau} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Matricea de rotație

$$\begin{bmatrix} x \cdot cos(\theta) - y \cdot sin(\theta) & x \cdot sin(\theta) + y \cdot cos(\theta) & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[x' \; y' \; 1] = [x \; y \; 1] \begin{bmatrix} \cos(u) & \sin(u) & 0 \\ -\sin(u) & \cos(u) & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ sau } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(u) & -\sin(u) & 0 \\ \sin(u) & \cos(u) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Matricile de simetrie (OX, OY, Origine, y=x)

$$[x'\ y'\ 1] = [x\ y\ 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ sau } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

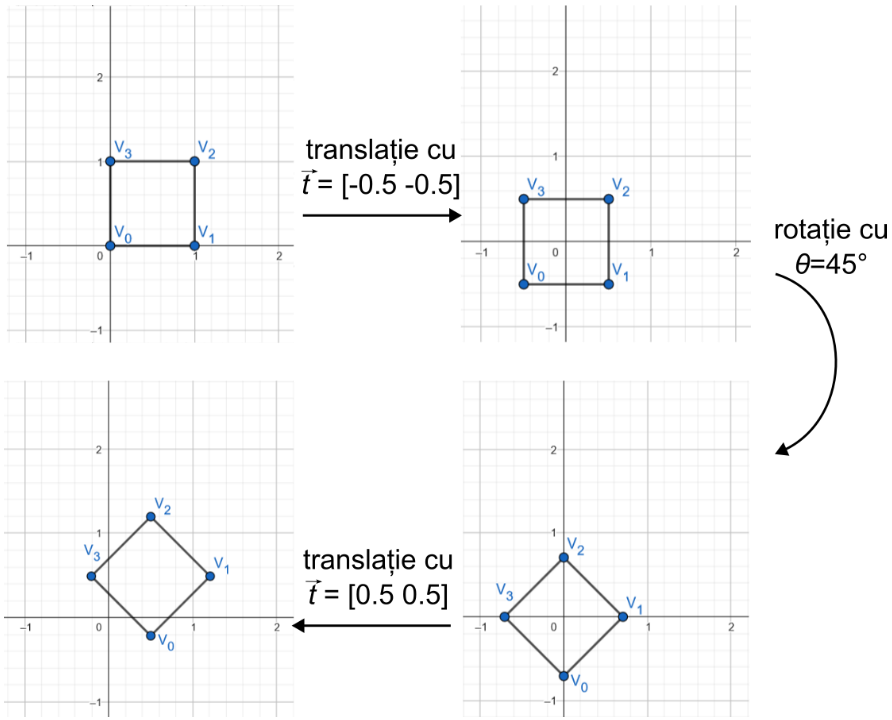
$$[x'\ y'\ 1] = [x\ y\ 1]\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ sau } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$[x'\ y'\ 1] = [x\ y\ 1]\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ sau } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$[x'\ y'\ 1] = [x\ y\ 1]\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ sau } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Compunerea tranformărilor

- Toate transformările prezentate anterior pot fi codificate folosind matrice de dimensiune $\mathbf{3} \times \mathbf{3}$
- Compunerea transformărilor se poate realiza prin înmulțirea matricilor corespunzătoare fiecărei transformări.



Compunerea tranformărilor - Exemplu

Scalarea față de un punct oarecare din plan.

Matricea transformarii compuse:

- Translaţia prin care punctul fix al transformării ajunge în origine: T(-xf, -yf);
- Scalarea faţă de origine: S(0,0,sx,sy);
- Translaţia inversă celei de la punctul 1: T(xf, yf).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T(xf, yf) * S(0,0,sx,sy)/R(0,0,u) * T(-xf, -yf) * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Cum arată lanțul de transformări 3D?

Matricea de translație

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matricea de modificare a scării

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matricile de rotație

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(u) & -sin(u) & 0 & 0 \\ sin(u) & cos(u) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformare este față de axa OZ.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(u) & 0 & sin(u) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(u) & 0 & cos(u) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformare este față de axa OY.

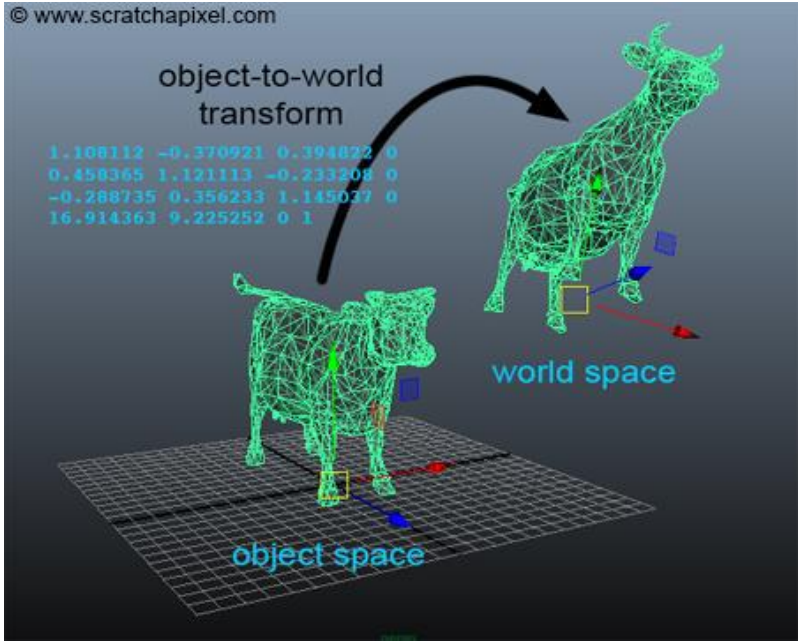
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(u) & -sin(u) & 0 \\ 0 & sin(u) & cos(u) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformare este față de axa OX.

Transformarea geometriei

Transformarea de modelare

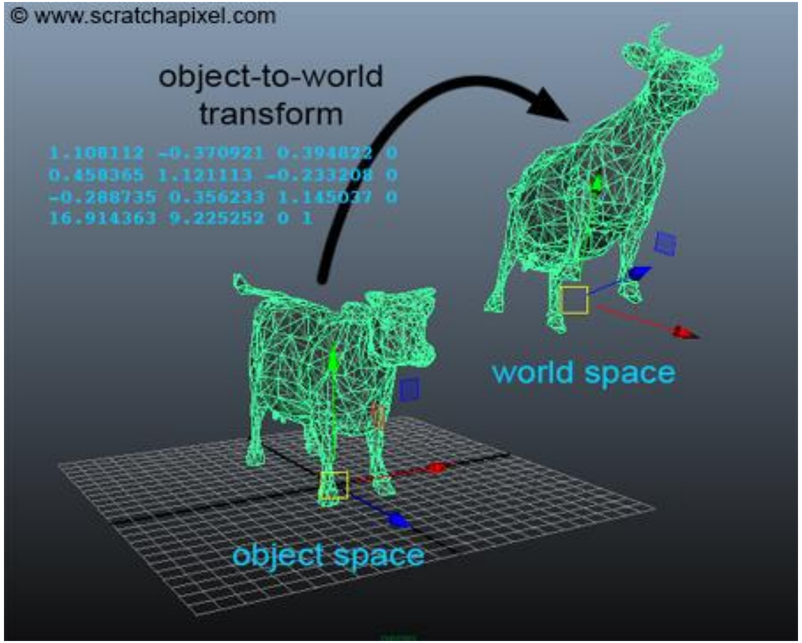
- Pozițiile vârfurilor sunt definite în Object Space (**Spațiul Obiect**), un spațiu în care toate pozițiile sunt relative la poziția obiectului.
- Pentru a poziționa un obiect în scenă se folosește alt spațiu, numit World Space (**Spațiul Scenei 3D**).



Transformarea de modelare

Transformarea de modelare:

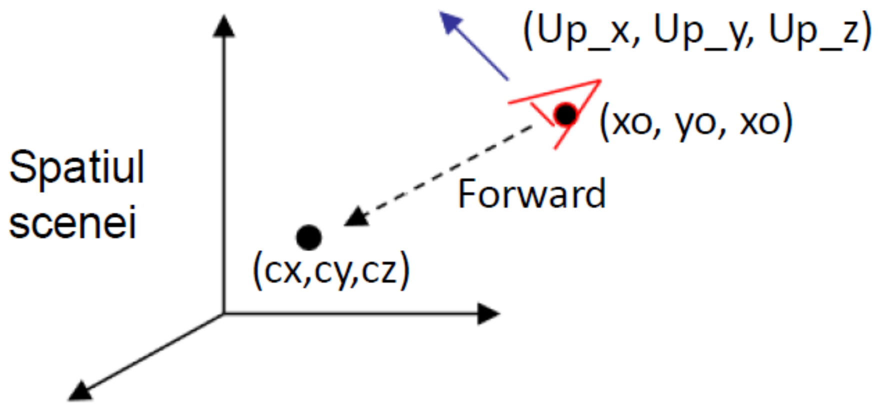
- Se iau toate transformările care se efectuează asupra obiectului, se calculează matricele acestora, iar apoi se înmulțesc. Rezultatul va fi o matrice numită **World Matrix**.
- Această matrice va fi înmulțită cu coordonatele fiecărui vârf al geometriei, astfel transformând coodronatele din **Object Space în World Space**.



Transformarea de vizualizare

- În scenă (presupunem că) există o **cameră** definită prin:
  - Poziție (xo, yo ,zo)
  - Orientare (vector Forward și vector Up).

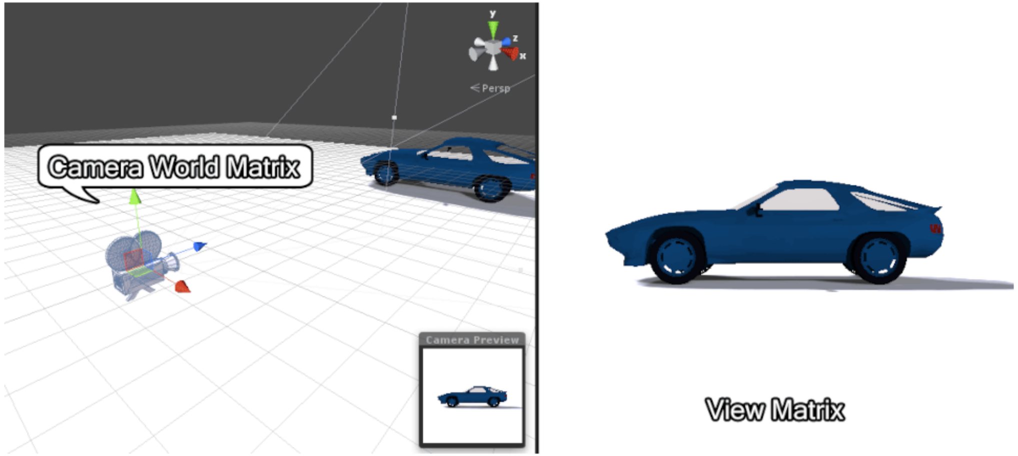
- Putem crea o **matrice M_{cam}** care să codifice transformarea camerei.



Transformarea de vizualizare

Transformarea de vizualizare:

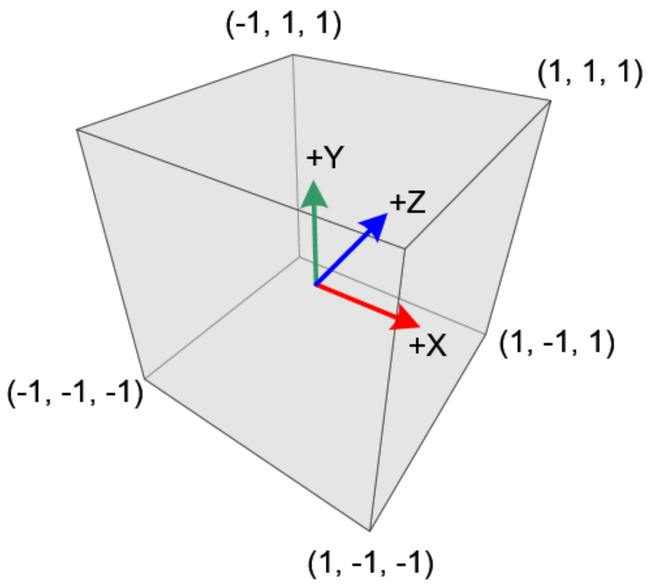
- Asupra geometriei aplicăm o transformare astfel încât să pară că aceasta este văzută din perspectiva camerei. Acest spațiu se numește View Space (**Spațiul Observator**).
- Pentru un punct P din World Space în View Space este nevoie să aplicăm transformarea $P' = M^{-1}{}_{cam} * P$.
- Matricea se numește **Matrice de Vizualizare.**

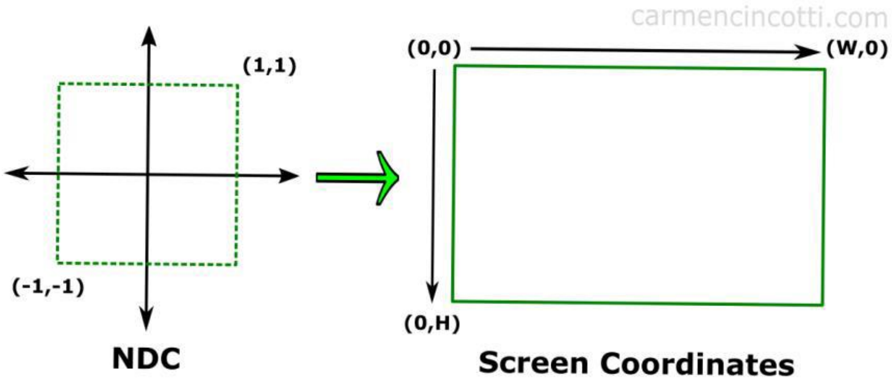


Normalized Display Coordinates (NDC)

- API-urile grafice folosesc coordonate într-un spațiu normalizat (coordonatele au valori între -1 și 1 sau între 0 și 1).
- În OpenGL, NDC-ul poate fi vizualizat ca un cub cu latura de lungime 2 și extremitățile (−1,−1,−1), respectiv (1,1,1).
- Acest spațiu de coordonate se numește Clip Space (**Spațiu de Decupare**) și reprezintă spațiul final în care vom dori să aducem coordonatele pentru ca geometria să fie desenată pe ecran.

Normalized Display Coordinates (NDC)

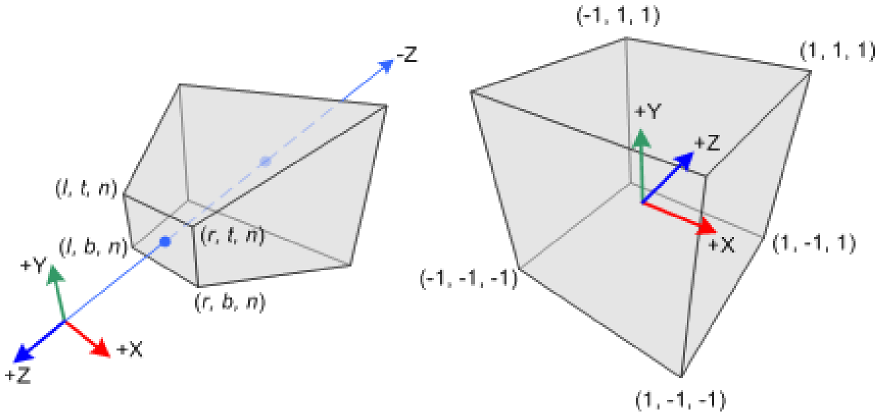




Transformarea de proiecție

Transformarea de proiecție:

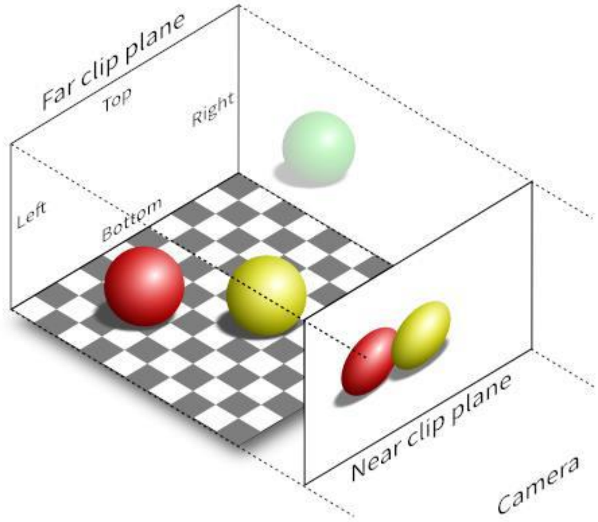
- Pentru trecerea de la View Space la Clip Space se folosește o matrice de 4x4 numită Projection Matrx (**Matrice de Proiecție**)
- În funcție de specificul aplicației, aceasta poate fi construită în mai multe moduri.



Transformarea de proiecție

Proiecția ortografică:

- Proporțiile obiectelor sunt păstrate, indiferent de poziția acestora.
- Liniile paralele rămân paralele.



Orthographic projection (O)

Matricea de proiecție ortografică

$$Par = \begin{bmatrix} \frac{2}{right - left} & 0 & 0 & -A \\ 0 & \frac{2}{top - bottom} & 0 & -B \\ 0 & 0 & \frac{-2}{far - near} & C \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

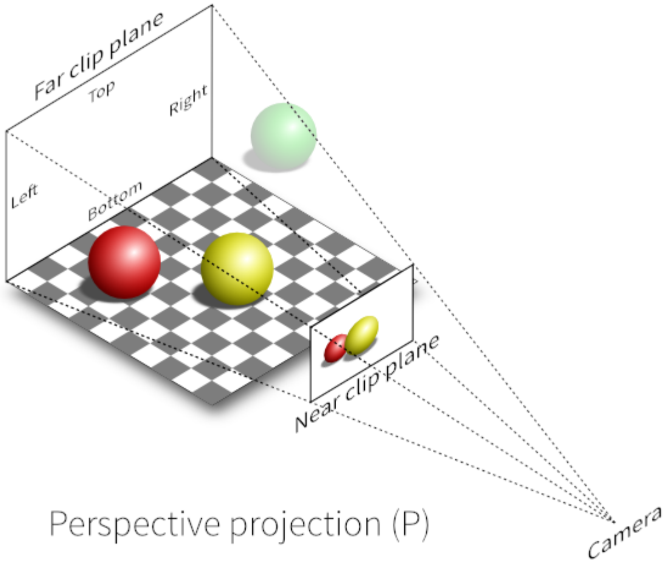
$$A = \frac{right + left}{right - left} \qquad B = \frac{top + bottom}{top - bottom}$$

$$C = -\frac{far + near}{far - near} \qquad D = -\frac{2 * far * near}{far - near}$$

Transformarea de proiecție

Proiecția perspectivă:

- Proiecție realistă, asemănătoare cu vederea umană
- Se calculează în funcție de:
    - FoV (Field of View),
    - Aspect ratio
    - Distanța minimă (near plane)
    - Distanța maximă (far plane)



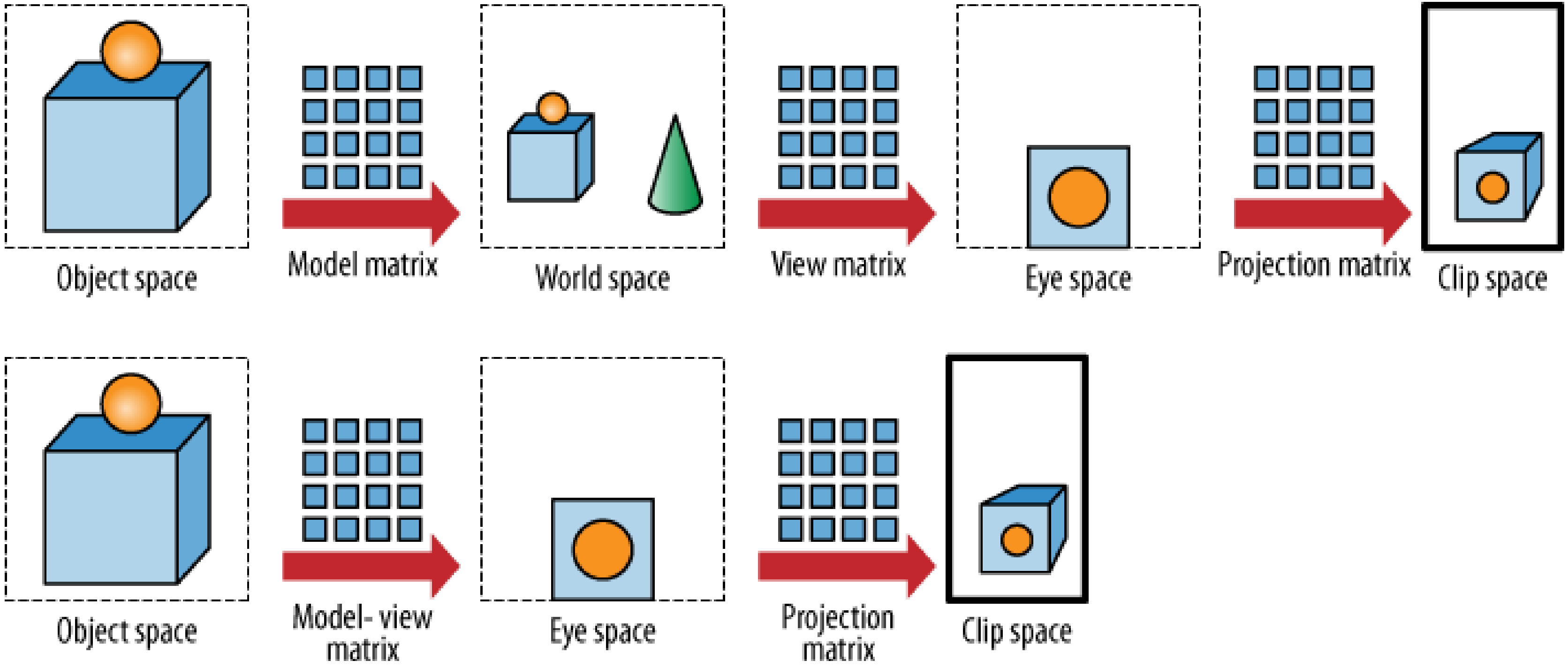
Perspective projection (P)

Matricea de proiecție perspectivă

$$
Per = \begin{bmatrix} \frac{2 * near}{right - left} & 0 & A & 0 \\ 0 & \frac{2 * near}{top - bottom} & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{bmatrix}
$$

$$A = \frac{right + left}{right - left} \qquad B = \frac{top + bottom}{top - bottom}$$

$$C = -\frac{far + near}{far - near} \qquad D = -\frac{2 * far * near}{far - near}$$



$$P_{clip}=M_{proj}M_{view}M_{world}P_{object}$$

https://www.oreilly.com/library/view/iphone-3d-programming/9781449388133/ch02.html

Concluzie

Spatiul obiect – Object space (Sistemul coordonatelor obiect)

Matricea M ↓ **Transformarea de modelare**

Spatiul scenei 3D – World space (Sistemul coordonatelor globale)

Matricea V ↓ **Transformarea de vizualizare**

Spatiul observator – Eye space (Sistemul coordonatelor observator)

Matricea P ↓ **Transformarea de proiectie (volumul vizual)**

Spatiul de decupare – Clip space (Sistemul coordonatelor de decupare)

Matricea MVP

↓ **Impărţirea perspectivă**

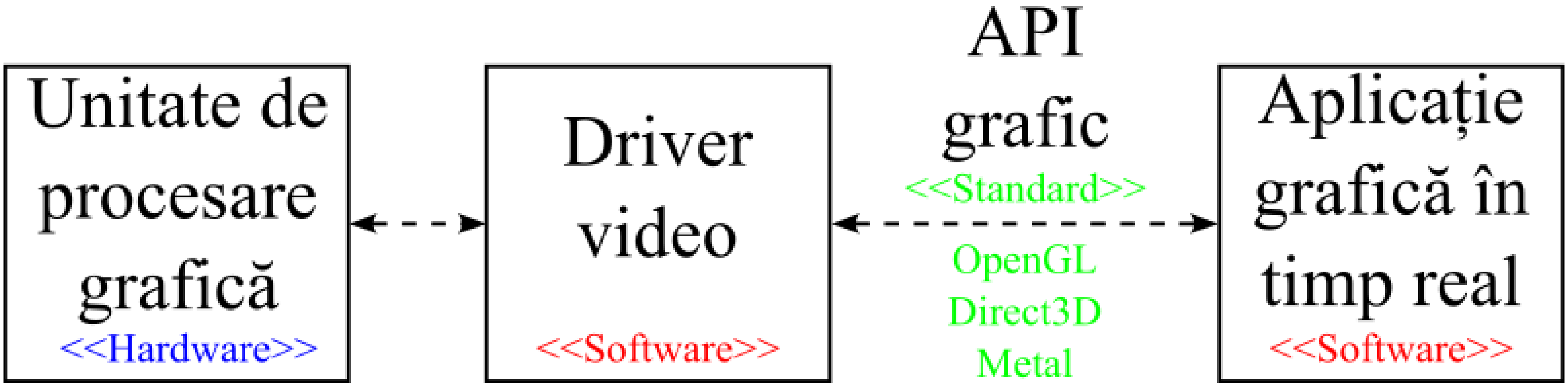
Sistemul coordonatelor dispozitiv normalizate

↓ **Transformarea în poarta de afişare (viewport)**

Sistemul coordonatelor ecran ataşat porţii de afişare

04

OpenGL Pipeline

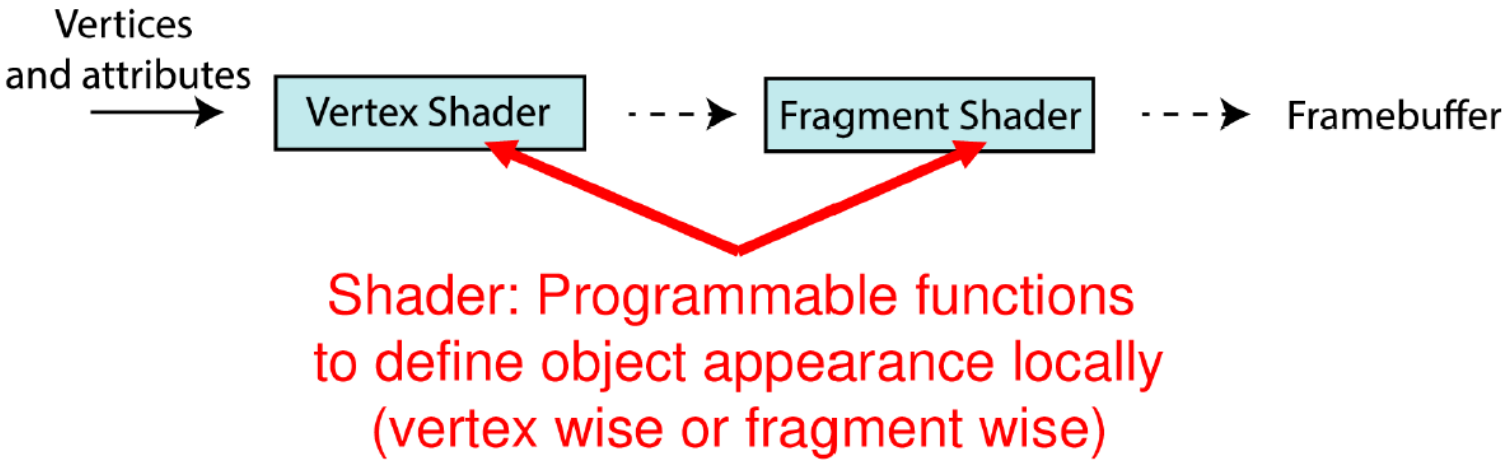


API-ul grafic OpenGL

World Coordinates

Pixel-wise attributes

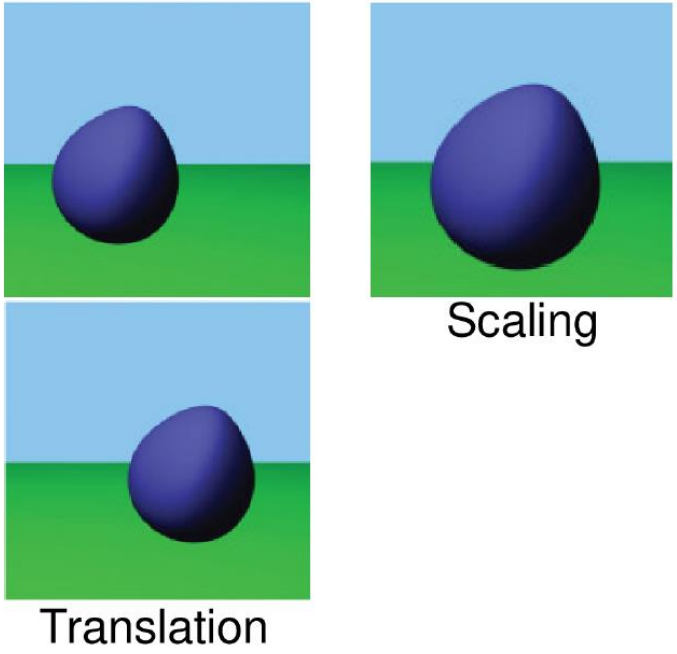
RGBA image



Randarea cu OpenGL

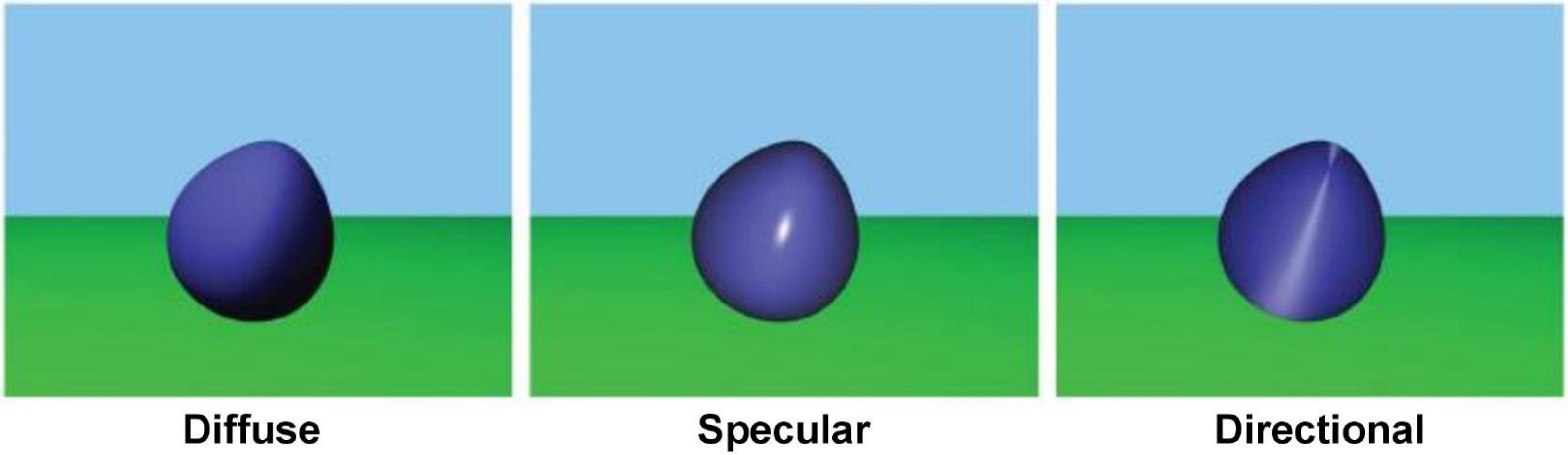
Vertex Shader

- Pentru fiecare vârf din lista de vârfuri, se execută o singură instanță de program de tip **Vertex Shader**, prelucrând informația fiecărui vârf în parte.
- Putem **aplica transformări** învățate anterior:
    - Scalare, Rotație, Translație, Simetrie etc.
    - Proiecții: ortografice, perspectivă



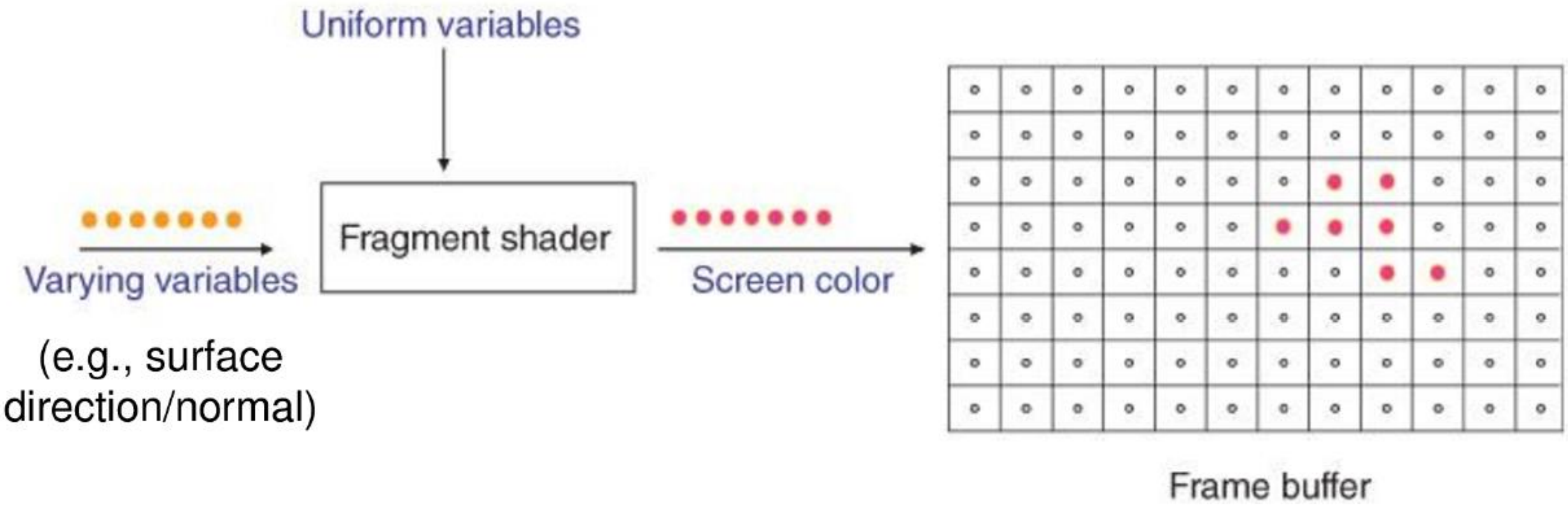
Fragment Shader

- Pentru fiecare pixel ce a fost procesat de către rasterizator și pentru care trebuie să i se atribuie o culoare, rasterizatorul execută o instanță a programului de tip **Fragment Shader** pentru prelucrarea de pixeli.
- Putem simula materiale, lumini, putem aplica texturi etc.



Fragment Shader

(e.g., light direction)



(e.g., surface direction/normal)

Resurse

“CPSC 427, Fall 2021.” https://www.cs.ubc.ca/%7Erhodin/2021_2022_CPSC_427/

“Learn OpenGL, extensive tutorial resource for learning Modern OpenGL.” https://learnopengl.com/

“3D math primer for graphics and game development.” https://gamemath.com/

Jeremiah, “Understanding the view matrix,” 3D Game Engine Programming, Feb. 06, 2018. https://www.3dgep.com/understanding-the-view-matrix/

Curs - „Grafică pe calculator” –UB, FMI, Informatică An 3

Curs – „Elemente de Grafică pe Calculator” – UPB, ACS

Curs – „Programarea prelucrărilor în banda grafică” – UPB, ACS