Reprezentarea obiectelor in scena 3D (matricea MVP)

Pentru a putea plasa obiectul in scena si apoi pentru a ne putea “deplasa” in scena creata trebuie sa avem in vedere 3 matrici:

- **Model** (notata *M*) – este matricea care cuprinde transformarile aplicate asupra obiectului: scalari, rotatii, translatii (in aceasta ordine). Rotatiile se aplica dupa regula roll-pitch-yaw (fata de axa Ox, fata de axa Oy, fata de axa Oz). Vedeti si: http://msl.cs.uiuc.edu/planning/node102.html
- **View** (notata *V*) – matricea care ne permite sa ne deplasam in scena, folosind punctul abstract numit *camera.* Practic daca vrem sa ne deplasam, de exemplu, la dreapta, atunci scena, relativ la noi se va deplasa la stanga. Aceasta deplasare va fi continuta de View.
- **Projection** (notata *P*) – matricea care proiecteaza obiectele in perspectiva (astfel incat obiectele mai apropiate apar in scena mai mari, iar cele mai departate apar mai mici).

Pentru a reprezenta obiectul in scena, vertecsii lui vor fi inmultiti cu matricea *MVP=M*V*P.*

Clasa Camera

Proprietati	**Metode**
• position (punct, Vector3 sau Vector4) • target (punct, Vector3 sau Vector4) • up (directie, Vector3 sau Vector4) • moveSpeed (GLfloat) • rotateSpeed (GLfloat) • near (GLfloat) • far (GLfloat) • fov (GLfloat) • deltaTime (GLfloat) • Eventual si xAxis, yAxis, zAxis (Vector3 sau Vector4), viewMatrix, worldMatrix (Matrix)	• moveOx() • moveOy() • moveOz() • rotateOx() • rotateOy() • rotateOz() • updateWorldView() • getteri si setteri • constructor(i)

Daca alegeti Vector4 pentru vectori va trebui uneori sa faceti conversia la Vector3 sau sa definiti o parte din operatii pentru Vector4. Daca alegeti Vector3 va trebui sa faceti conversia la Vector4 cand inmultiti cu matricile de transformare.

Proprietatea deltaTime va fi setata in update (va reprezenta timpul care a trecut intre doua update-uri si deci intre doua frame-uri).

zAxis = -(target - position).Normalize();

yAxis = up.Normalize();

xAxis = zAxis.Cross(yAxis).Normalize();

Miscare inainte

*forward = -(target - position).Normalize() * directie;*

*vectorDeplasare = forward * moveSpeed * deltaTime;*

position += vectorDeplasare;

target += vectorDeplasare;

Rotatie fata de OY

localTarget = Vector4(0.0f, 0.0f, -(target - position).Length(), 1.0f);

*rotatedTarget = localTarget * mRotateOY;*

*target = rotatedTarget * worldMatrix;*

Rotatie fata de OX

*rotatedLocalUp = localUp * mRotateOX;*

*up = rotatedLocalUp * worldMatrix*

up = up.Normalize();

localTarget = Vector4(0.0f, 0.0f, -(target - position).Length(), 1.0f);

*rotatedTarget = localTarget * mRotateOX;*

*target = rotatedTarget * worldMatrix*

Matricile worldMatrix si viewMatrix

*worldMatrix = R*T*

unde T reprezinta matricea de transaltie pentru vectorul (position.x, position.y, position.z);

viewMatrix = $worldMatrix^{-1}$

R – matrice ortogonala => $R^{-1} = R^t$

viewMatrix= T^{-1} * R^t

R^t = *R* transpus

iar T^{-1} e chiar translatia cu vectorul (-position.x, -position.y, -position.z)

$$R=\begin{pmatrix} xAxis.x & xAxis.y & xAxis.z & 0 \\ yAxis.x & yAxis.y & yAxis.z & 0 \\ zAxis.x & zAxis.y & zAxis.z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

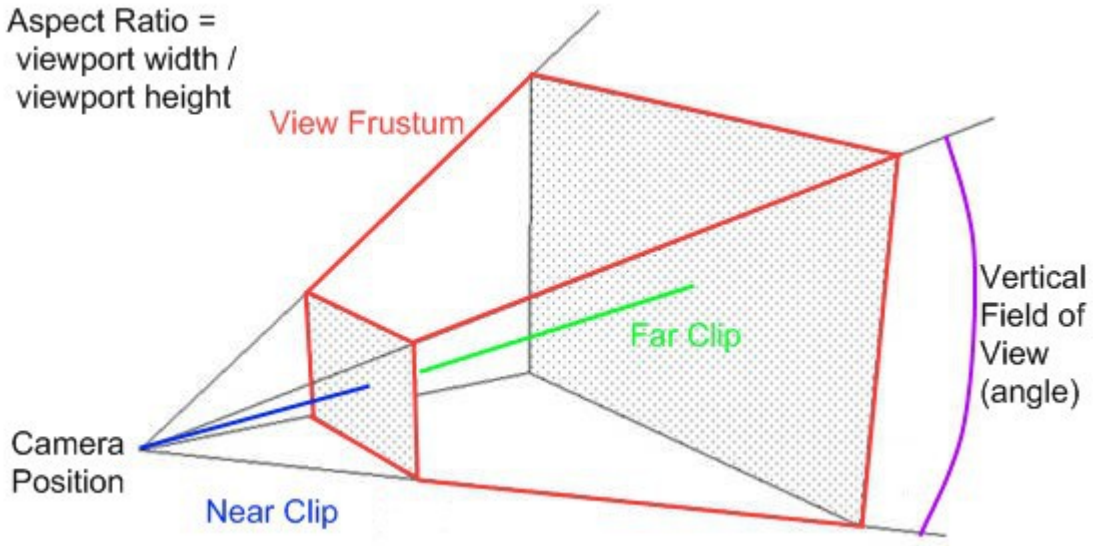
Proiectia

Tipurile de proiectii cu care vom lucra sunt:

- **proiectia in perspectiva** (folosita pentru obiectele in scena 3D). Folositi setPerspective() din Math.h.

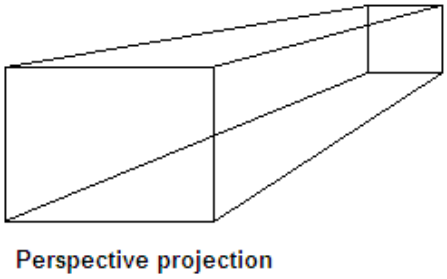
  Avem 3 date importante pentru proiectia in perspectiva:

  - **fov** (field of view) – deschiderea trunchiului de piramida (unghiul din varf)
  - planul **near** (baza mica a trunchiului de piramida; planul mai apropiat de *camera*).
  - planul f**ar** (baza mare a trunchiului de piramida; planul mai departat de *camera*).
  - **Aspectul** (aspect ratio) = widthFerestra/heightFereastra

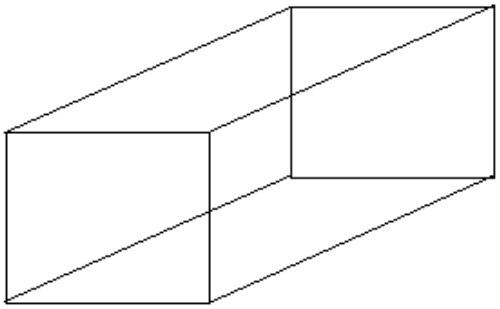


(imagine preluata de la: https://ksgamedev.wordpress.com/tag/camera/)

- **proiectia ortografica, numita si ortogonala** (folosita in general pentru ecrane 2D – cum ar fi ecrane de setari, meniuri etc)



Perspective projection



Orthographic projection

(imagine preluata de la: http://gamedev.stackexchange.com/questions/76111/open-gl-perspective-projection-vs-orthographic-projection)

Cititi si:

- http://www.scratchapixel.com/lessons/3d-basic-rendering/perspective-and-orthographic-projection-matrix/building-basic-perspective-projection-matrix
- https://wiki.blender.org/index.php/Doc:RO/2.4/Manual/3D_interaction/Navigating/3D_View (aici gasiti si o imagine care compara cele doua tipuri de proiectii)