Afisarea obiectelor 3D

Operatii asupra varfurilor:

- Transformari geometrice
- Proiectia 3D-2D
- Transformarea fereastra-poarta asupra varfurilor 2D

Afisare:

-Wire-frame (prin laturi)→ reprezentare prin coordonate varfuri si laturi(legaturi intre varfuri)

-Cu fețe opace→ reprezentare prin:

- coordonate varfuri
- contururi fețe (cicluri de varfuri): topologia obiectului

Reprezentarea obiectelor 3D

```
class Face
{
public:
        vector<int> contour;
public:
        Face(vector <int> list)
        {   for (int i = 0; i < list.size(); i++)
                                contour.push_back(list[i]);
        }
};
```

```
class Object3D
{
public:
    vector<Point3D*> vertices;
    vector<Point3D*> transf_vertices;
    vector<Face*> faces;
    Color color;
    bool fill; //tipul de interior
.....
};
```

```
class PolyLine3D:public Object3D
{
public:
    PolyLine3D(vector <Point3D*> listV,Color _color)
    {
    .....
                color = _color;
                fill = false;
    }
};
```

```
class Transform3D
{
public:
	static float ModelMatrix[4][4]; // matrice in care se acumuleaza transformarile geometrice succesive
	static float ProjectionMatrix[4][4]; // matricea de proiectie
	static float MVPMatrix[4][4]; // acumuleaza transformarea de modelare cu cea de proiectie
	static bool proj_type; //false=proiectie paralela, true=proiectie perspectiva

	static void loadIdentityModelMatrix(); //matricea curenta de modelare devine matricea identitate
	static void multiplyModelMatrix(float matrix[4][4]); //inmulteste matricea matrix cu matricea de
	modelare
	static void translateMatrix(float tx, float ty, float tz);
	static void scaleMatrix(float sx,float sy,float sz);
	static void rotateMatrixOx(float u);
	static void rotateMatrixOy(float u);
	static void rotateMatrixOz(float u);
	static void applyTransform(Object3D *o); // calculeaza MVPMatrix si o aplica varfurilor obiectului
	static void applyTransform(Point3D *p,Point3D *transf_p); // aplica MVPMatrix unui singur varf

	static void parallelProjectionMatrix(float a, float b, float c); // calculeaza matricea proiectiei paralele
	                                                                 // si o memoreaza in ProjectionMatrix
	static void perspectiveProjectionMatrix(float xc, float yc, float zc); // calculeaza matricea proiectiei
	                                                                       // perspectiva si o memoreaza in ProjectionMatrix
	 static void  obliqueProjectionMatrix (float f, float alfa);
	static void axonometricProjectionMatrix(float ux, float uy);
};
```