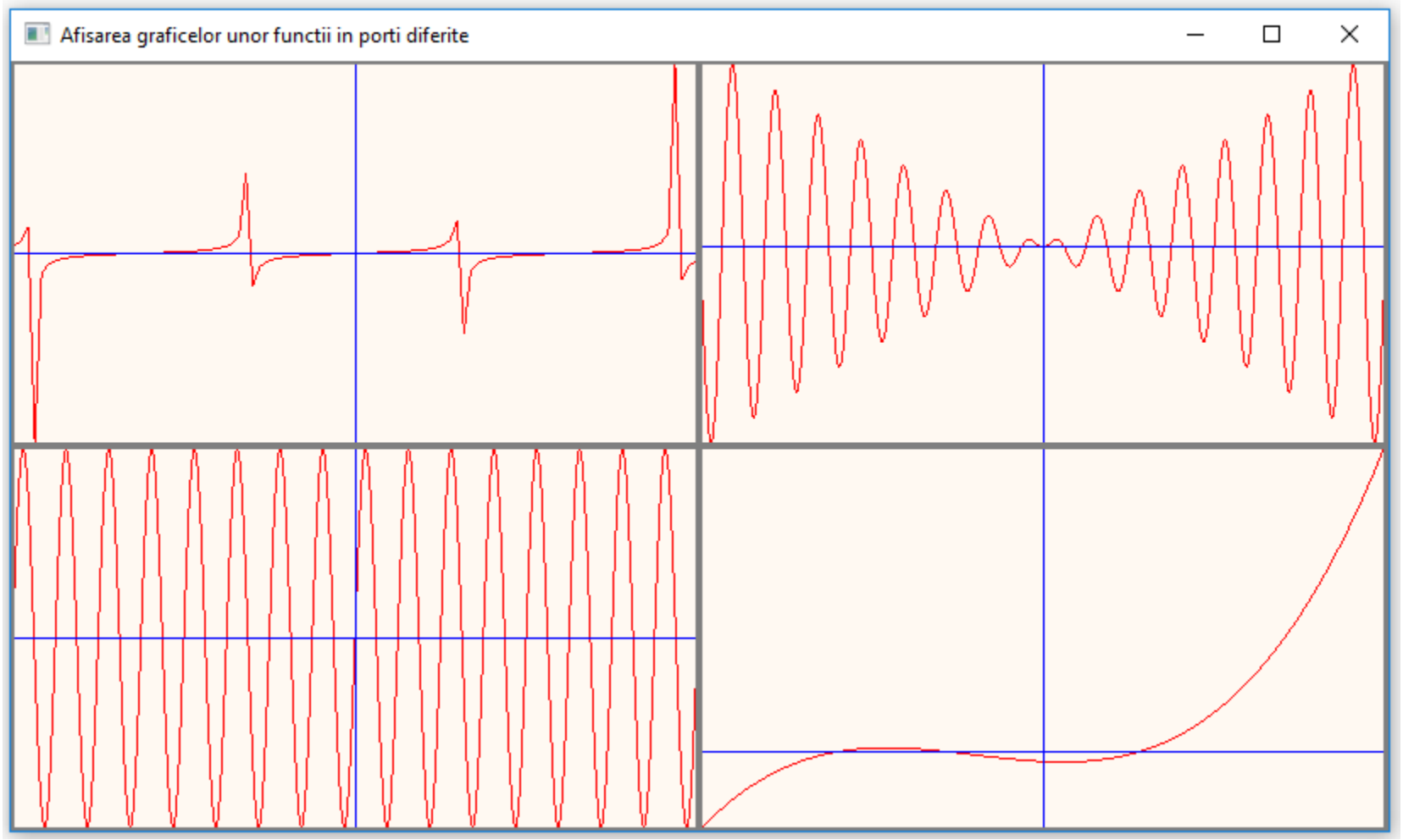
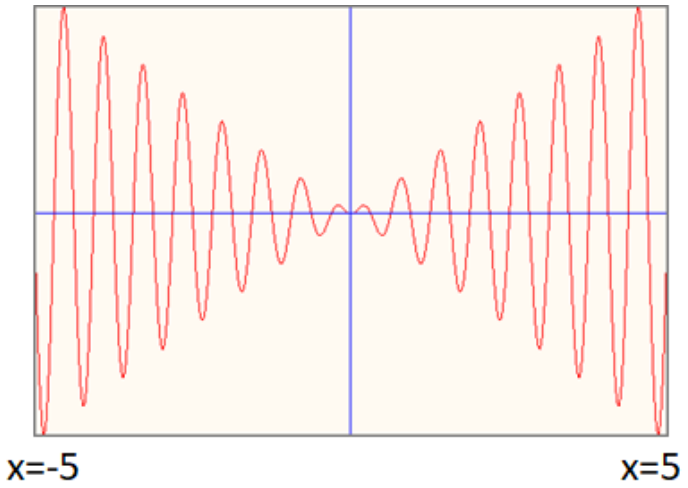
Afişarea graficelor unor funcţii în porţi diferite

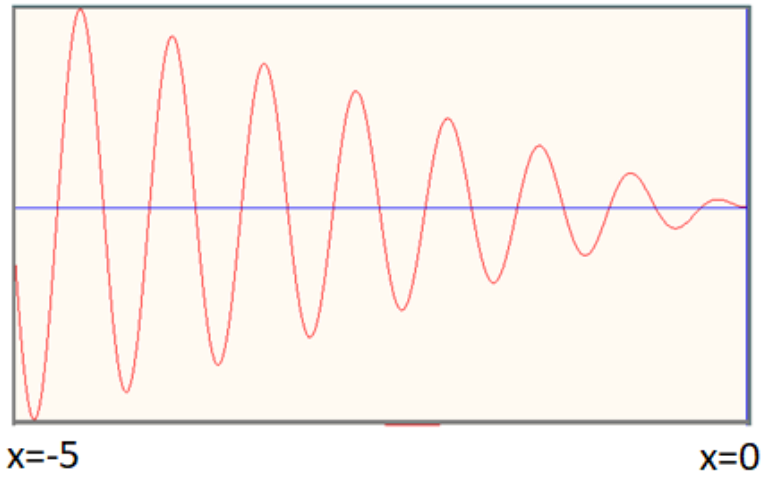


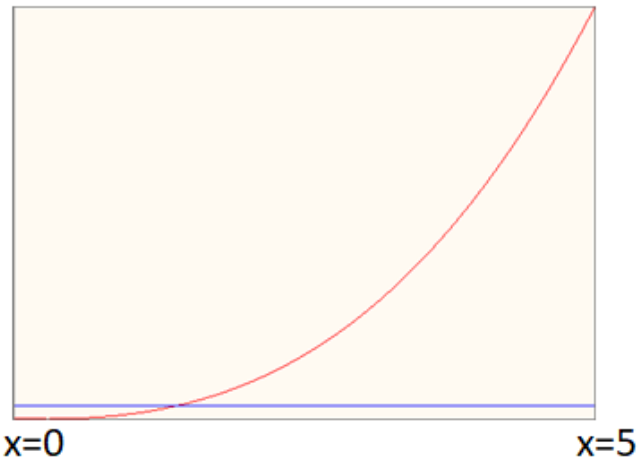
Object2DGrafic.cpp

```
Mesh* Object2DGrafic::CreateFunctionPlot(std::string name,
        float &x1, float &x2, float &y1, float &y2, float pas, float(*f)(double f))
```

//Creaza graficul unei functii matematice y=f(x), pe un interval x1-x2 dat, în spatiul logic
//Include in desen si axele sistemului de coordonate







CreateFunctionPlot (1)

```
Mesh* Object2DGrafic::CreateFunctionPlot(std::string name,
        float &x1, float &x2, float &y1, float &y2, float pas, float(*f)(double f))
{   // calculeaza punctele de pe graficul functiei matematice y = f(x) in intervalul x1 – x2,
    //in coordonate logice
   // întoarce în (x1,y1), (x2,y2) încadrarea graficului incluzand si axele sistemului de coordonate


// calculeaza incadrarea graficului pe axa OY,  [y1-y2], in intervalul xmin - xmax
     float x, y, xmin=x1, xmax=x2;
     y1 = y2 = f(xmin);
     for (x = xmin + pas; x<xmax; x += pas)
       {
             if ((y = f(x))<y1) y1 = y;
             if (y>y2)y2 = y;
       }
     if ((y = f(xmax))<y1) y1 = y;
     if (y>y2) y2 = y;
```

CreateFunctionPlot (2)

```
// Include axele OX, OY în încadrarea graficului
   if (0<xmin)x1 = 0; else x1 = xmin;
   if (0<y1)y1 = 0;
   if (0>xmax)x2 = 0; else x2 = xmax;
   if (0>y2)y2 = 0;

Mesh* plot = new Mesh(name);
plot->SetDrawMode(GL_LINES); // se vor uni câte 2 varfuri succesive
std::vector<VertexFormat> vertices;
std::vector<unsigned short> indices;
// memoreaza varfurile axei OX – se asociaza varfurilor culoarea albastru
    vertices.push_back(VertexFormat(glm::vec3(x1, 0, 0), glm::vec3(0, 0, 1)));
    vertices.push_back(VertexFormat(glm::vec3(x2, 0, 0), glm::vec3(0, 0, 1)));
// memoreaza varfurile axei OY – se asociaza varfurilor culoarea albastru
   vertices.push_back(VertexFormat(glm::vec3(0, y1, 0), glm::vec3(0, 0, 1)));
   vertices.push_back(VertexFormat(glm::vec3(0, y2, 0), glm::vec3(0, 0, 1)));
   indices.push_back(0); indices.push_back(1); indices.push_back(2);indices.push_back(3);
```

CreateFunctionPlot (3)

```
// se adauga varfurile care aproximeaza graficul;
// se asociaza varfurilor culoarea roşu
   int contor = 4; //nr de indici in buffer-ul de indici
   for (x = xmin; x<xmax - pas; x += pas)
    { vertices.push_back(VertexFormat(glm::vec3(x, f(x), 0), glm::vec3(1, 0, 0)));
      indices.push_back(contor); indices.push_back(contor+1);
      contor++;
     }
// se adauga ultimul varf
    vertices.push_back(VertexFormat(glm::vec3(xmax, f(xmax), 0), glm::vec3(1, 0, 0)));

    plot->InitFromData(vertices, indices);
   return plot;

}
```

TransformareaFereastraPoartaGrafic.cpp

```
void TransformareaFereastraPoartaGrafic::Init()
{
//setare camera
…….

// se creaza graficele celor 4 functii in intervalul xmin=-5, xmax=5
        float y1, y2, x1=-5.0f, x2 = 5.0f;
// se creaza un mesh cu graficul functiei f1; se seteaza spatiul logic pentru graficul functiei f1
        Mesh* mesh_f1 = Object2DGrafic::CreateFunctionPlot("mesh_f1", x1, x2, y1, y2, 0.01f, f1);
        AddMeshToList(mesh_f1);
// Seteaza spatial logic folosind valorile intoarse de CreateFunctionPlot in x1, y1, x2,y2
        logicSpace1.x = x1;
        logicSpace1.width = x2 - x1;
        logicSpace1.y = y1;
        logicSpace1.height = y2 - y1;
//se creaza, in mod analog, mesh_f2, mesh_f3, mesh_f4 şi logicSpace2, logicSpace3, logicSpace
……..
}
```

TransformareaFereastraPoartaGrafic.cpp

```
void TransformareaFereastraPoartaGrafic::Update(float deltaTimeSeconds)
{
    glm::ivec2 resolution = window->GetResolution(); // fereastra aplicatiei

// seteaza zona de desenare – partea stanga-sus
   viewSpace = ViewportSpace(0, resolution.y / 2, resolution.x / 2, resolution.y / 2);
   SetViewportArea(viewSpace, glm::vec3(1, 0.98, 0.95), true); //culoarea fondului pt viewport: bej
   visMatrix = TransfVis2D(logicSpace2, viewSpace);
   DrawFunctionPlot(visMatrix, 2);//deseneaza gaficul functiei f2
// partea dreapta-sus a ferestrei aplicatiei
   viewSpace = ViewportSpace(resolution.x / 2, resolution.y / 2, resolution.x / 2, resolution.y / 2);
   SetViewportArea(viewSpace, glm::vec3(1,0.98,0.95), true);
   visMatrix = TransfVis2D(logicSpace1, viewSpace);
   DrawFunctionPlot(visMatrix, 1); //deseneaza graficul functiei f1
// analog pentru desenarea celorlalte functii
}
```

TransformareaFereastraPoartaGrafic.cpp

```
void TransformareaFereastraPoartaGrafic::DrawFunctionPlot(glm::mat3 visMatrix, int nr_functie)
{
    modelMatrix = visMatrix;
    switch (nr_functie)
    {
     case 1:
        RenderMesh2D(meshes["mesh_f1"],shaders["ShaderGrafic"],modelMatrix);
        break;
     case 2:
        RenderMesh2D(meshes["mesh_f2"], shaders["ShaderGrafic"], modelMatrix);
        break;
     case 3:
        RenderMesh2D(meshes["mesh_f3"], shaders["ShaderGrafic"], modelMatrix);
        break;
     case 4:
        RenderMesh2D(meshes["mesh_f4"], shaders["ShaderGrafic"], modelMatrix);
        break;
     default:
        break;
    }
}
```