Băicoianu Bianca
Grupa 351

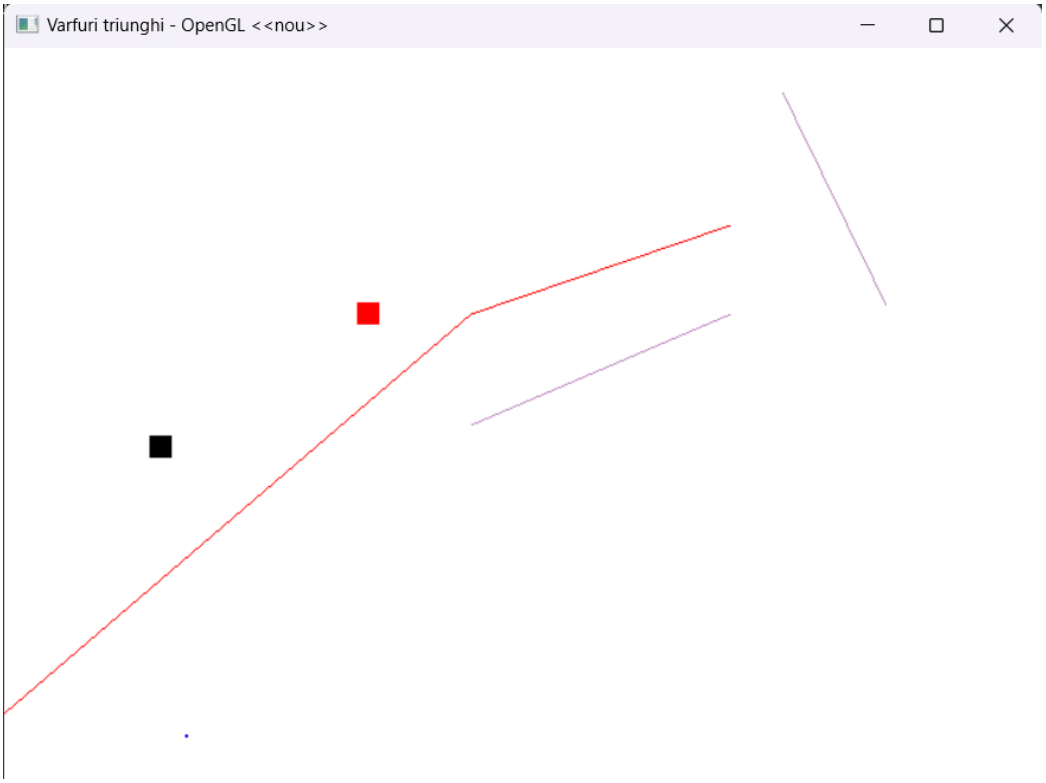
Laborator 1

- Exercițiul 1&2
  - rezolvate la laborator 04.10.2023

- Exercițiul 3

*3) **(1p)** Transferati unul dintre codurile sursa 01_03_puncte_segmente_OLD.cpp sau 01_04_poligoane_OLD.cpp in versiunea "noua" a OpenGL. Imaginea nu trebuie sa fie exacta (la scara), ci aproximativa*

Rezultat:



01_03_puncte_segmente.cpp

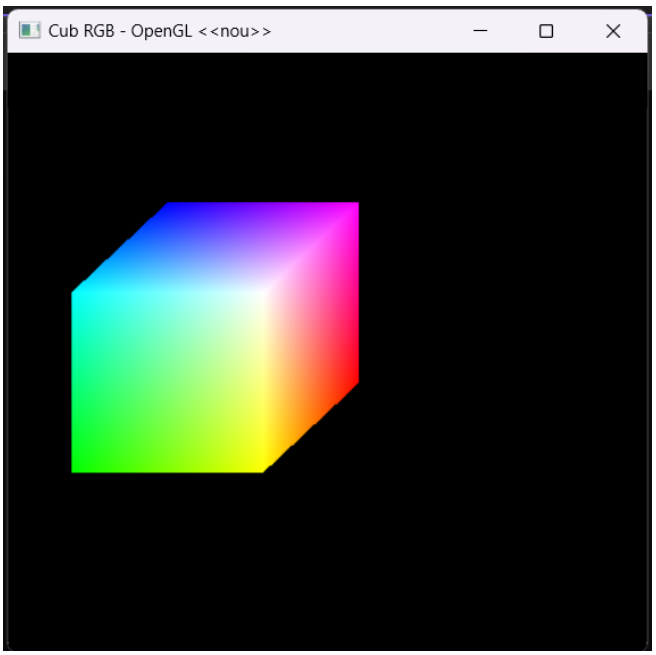
```
01_03_puncte_segmente.cpp*
lab1                                                    (Global Scope)
void CreateVBO(void)
{
    //  Coordonatele varfurilor;
    GLfloat Vertices[] = {

        -0.7f, 0.1f, 0.0f, 1.0f, // pct negru
         -0.3f, 0.4f, 0.0f, 1.0f,  // pct rosu
         -1.0f, -0.5f, 0.0f, 1.0f,  //primul segm rosu
         -0.1f, 0.4f, 0.0f, 1.0f,
          0.4f, 0.6f, 0.0f, 1.0f, //al doilea segm rosu
          -0.1f,0.15f,0.0f, 1.0f, // primul segm mov
          0.4f, 0.4f, 0.0f, 1.0f,
          0.7f, 0.42f, 0.0f, 1.0f, // al doilea segm mov
          0.5f, 0.9f, 0.0f, 1.0f,
          -0.84f, -0.75f, 0.0f, 1.0f, //"segm" albastra
          -0.85f, -0.76f, 0.0f, 1.0f,
          -0.65f, -0.55f, 0.0f, 1.0f, //pct albastru
    };

    //  Culorile in spectrul RGB ca atribute ale varfurilor;
    GLfloat Colors[] = {
        0.0f, 0.0f, 0.0f, 1.0f, // negru
        1.0f, 0.0f, 0.0f, 1.0f, // rosu
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        0.74f, 0.54f, 0.74f, 1.0f, //mov
        0.74f, 0.54f, 0.74f, 1.0f,
        0.74f, 0.54f, 0.74f, 1.0f,
        0.74f, 0.54f, 0.74f, 1.0f,
        0.0f, 0.0f, 1.0f, 1.0f, // albastru
        0.0f, 0.0f, 1.0f, 1.0f,
        0.0f, 0.0f, 1.0f, 1.0f

    };
```

```
//  Functia de desenarea a graficii pe ecran;
void RenderFunction(void)
{
    glClear(GL_COLOR_BUFFER_BIT);       //  Se curata ecranul OpenGL pentru a fi desenat noul continut;
    glPointSize(15.0);                  //  Se seteaza dimensiunea punctelor;

    //  Desenarea primitivelor geometrice - varfurile unui triunghi, utilizand datele din VBO
    //  Functia primeste 3 argumente:
    //  - arg1 = tipul primitivei desenate,
    //  - arg2 = indicele primului punct de desenat din buffer,
    //  - arg3 = numarul de puncte consecutive de desenat;

    glDrawArrays(GL_POINTS, 0, 2);  // punctele
    glDrawArrays(GL_LINE_STRIP, 2, 3);  // segmentele rosii
    glDrawArrays(GL_LINES, 5, 4);   // segmentele mov
    glDrawArrays(GL_LINES, 9, 2);   // "linie albastra"
    glPointSize(2.0);
    glDrawArrays(GL_POINTS, 11, 1);


    glFlush();                      //  Asigura rularea tuturor comenzilor OpenGL apelate anterior;
}
```

- Exercițiul 4

*4) **(1p)** Realizati o reprezentare 2D simplificata a cubului RGB (puteti utiliza si aceasta resursa)*

Rezultat:



01_05_cubRGB.cpp

Principiu: fiecare față vizibilă în reprezentarea 2D a cubului a fost împărțită în 2 triunghiuri (am folosit ca reper a doua resursă din cerință)

```
//  Coordonatele varfurilor;
GLfloat Vertices[] = {

    -0.8f, 0.2f, 0.0f, 1.0f, //cyan
    -0.8f, -0.4f, 0.0f, 1.0f,  //green
    -0.2f,  0.2f, 0.2f, 1.0f, //white
    -0.2f, -0.4f, 0.0f, 1.0f, //yellow
    0.1f, 0.5f, 0.5f, 1.0f, //magenta
    0.1f, -0.1f, 0.3f, 1.0f,//red

    0.1f, 0.5f, 0.5f, 1.0f, //magenta
    -0.5f, 0.5f, 0.5f, 1.0f, // blue
    -0.2f,  0.2f, 0.2f, 1.0f, //white
    -0.8f, 0.2f, 0.0f, 1.0f, //cyan
};

//  Culorile in spectrul RGB ca atribute ale varfurilor;
GLfloat Colors[] = {

   0.0f, 1.0f, 1.0f, 1.0f, // cyan
   0.0f, 1.0f, 0.0f, 1.0f, // green
   1.0f, 1.0f, 1.0f, 1.0f, // white
   1.0f, 1.0f, 0.0f, 1.0f, // yellow
   1.0f, 0.0f, 1.0f, 1.0f, //magenta
   1.0f, 0.0f, 0.0f, 1.0f,  //red

   1.0f, 0.0f, 1.0f, 1.0f, //magenta
   0.0f, 0.0f, 1.0f, 1.0f, // blue
   1.0f, 1.0f, 1.0f, 1.0f, // white
   0.0f, 1.0f, 1.0f, 1.0f, // cyan
```

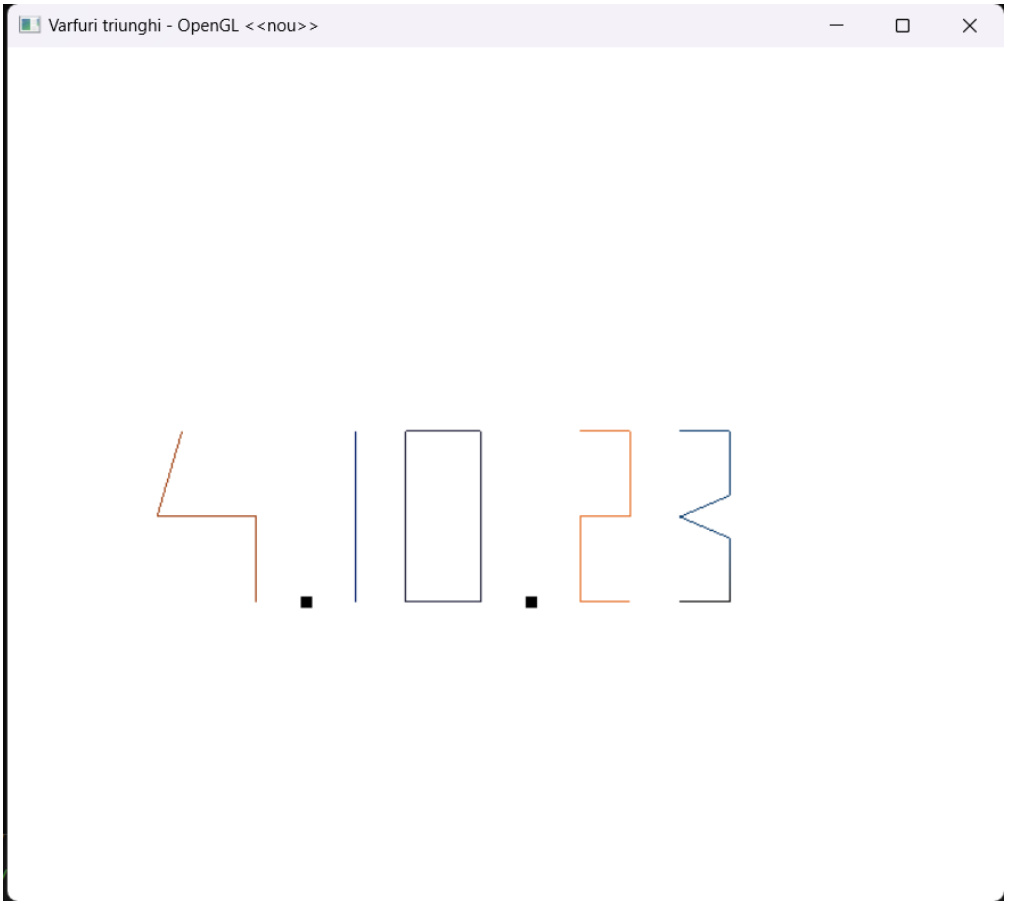
01_04_cubRGB.cpp

lab1 (Global Scope)

```
    CreateShaders();                        //  Initilizarea shaderelor;
}

//  Functia de desenarea a graficii pe ecran;
void RenderFunction(void)
{
    glClear(GL_COLOR_BUFFER_BIT);       //  Se curata ecranul OpenGL pentru a fi desenat noul continut;
    glPointSize(20.0);                  //  Se seteaza dimensiunea punctelor;

    //  Desenarea primitivelor geometrice - varfurile unui triunghi, utilizand datele din VBO
    //  Functia primeste 3 argumente:
    //  - arg1 = tipul primitivei desenate,
    //  - arg2 = indicele primului punct de desenat din buffer,
    //  - arg3 = numarul de puncte consecutive de desenat;

    glDrawArrays(GL_TRIANGLES, 0, 3); // triunghi cyan-green-white
    glDrawArrays(GL_TRIANGLES, 1, 3); // triunghi green-white-yellow
    glDrawArrays(GL_TRIANGLES, 2, 3); //triunghi white-yellow-magenta
    glDrawArrays(GL_TRIANGLES, 3, 3); //triunghi yellow-magenta-red
    glDrawArrays(GL_TRIANGLES, 6, 3); //triunghi magenta-blue-white
    glDrawArrays(GL_TRIANGLES, 7, 3); //triunghi blue-white-cyan


    glFlush();                          //  Asigura rularea tuturor comenzilor OpenGL apelate anterior;
}
```

- Exercițiul 5

*5) **(1p)** Folositi segmente de dreapta pentru a afisa data / alt text.*



01_05_data.cpp

lab1

```
//  Coordonatele varfurilor;
GLfloat Vertices[] = {
    // 4
    -0.65f, 0.1f, 0.0f, 1.0f,
     -0.7f, -0.1f, 0.0f, 1.0f,
     -0.5f, -0.1f, 0.0f, 1.0f,
     -0.5f, -0.3f, 0.0f, 1.0f,
     // punct
     -0.4f, -0.3f, 0.0f, 1.0f,
     // 1
      -0.3f, 0.1f, 0.0f, 1.0f,
      -0.3f,-0.3f,0.0f, 1.0f,
      // 0
      -0.2f, 0.1f, 0.0f, 1.0f,
      -0.05f, 0.1f, 0.0f, 1.0f,
      -0.05f, -0.3f, 0.0f, 1.0f,
      -0.2f, -0.3f, 0.0f, 1.0f,
      //punct
      0.05f, -0.3f, 0.0f, 1.0f,
      // 2
      0.15f, 0.1f, 0.0f, 1.0f,
      0.25f, 0.1f, 0.0f, 1.0f,
      0.25f, -0.1f, 0.0f, 1.0f,
      0.15f, -0.1f, 0.0f, 1.0f,
      0.15f, -0.3f, 0.0f, 1.0f,
      0.25f, -0.3f, 0.0f, 1.0f,
      // 3
      0.35f, 0.1f, 0.0f, 1.0f,
      0.45f, 0.1f, 0.0f, 1.0f,
      0.45f, -0.05f, 0.0f, 1.0f,
      0.35f, -0.1f, 0.0f, 1.0f,
      0.45f, -0.15f, 0.0f, 1.0f,
      0.45f, -0.3f, 0.0f, 1.0f,
      0.35f, -0.3f, 0.0f, 1.0
```

20 %
50 %
70 %
100 %
150 %
200 %
400 %

01_05_data.cpp

lab1

```
//  Culorile in spectrul RGB ca atribute
GLfloat Colors[] = {
    0.6f, 0.2f, 0.0f, 1.0f,
    0.6f, 0.2f, 0.0f, 1.0f,
    0.6f, 0.2f, 0.0f, 1.0f,
    0.6f, 0.2f, 0.0f, 1.0f,

    0.0f, 0.0f, 0.0f, 1.0f, // negru

    0.0f, 0.1f, 0.4f, 1.0f,
    0.0f, 0.1f, 0.4f, 1.0f,

    0.1f, 0.1f, 0.2f, 1.0f,
    0.1f, 0.1f, 0.2f, 1.0f,
    0.1f, 0.1f, 0.2f, 1.0f,
    0.1f, 0.1f, 0.2f, 1.0f,

    0.0f, 0.0f, 0.0f, 1.0f,// negru

    0.9f, 0.4f, 0.1f, 1.0f,
    0.9f, 0.4f, 0.1f, 1.0f,
    0.9f, 0.4f, 0.1f, 1.0f,
    0.9f, 0.4f, 0.1f, 1.0f,
    0.9f, 0.4f, 0.1f, 1.0f,
    0.9f, 0.4f, 0.1f, 1.0f,

    0.0f, 0.2f, 0.4f, 1.0f,
    0.0f, 0.2f, 0.4f, 1.0f,
    0.0f, 0.2f, 0.4f, 1.0f,
    0.0f, 0.2f, 0.4f, 1.0f,
    0.0f, 0.2f, 0.4f, 1.0f
};

//  Transmiterea datelor prin buffere;
```

```
//  Functia de desenarea a graficii pe ecran;
void RenderFunction(void)
{
    glClear(GL_COLOR_BUFFER_BIT);       //  Se curata ecranul OpenGL pentru a fi desenat noul continut;
    glPointSize(8.0);                   //  Se seteaza dimensiunea punctelor;

    //  Desenarea primitivelor geometrice - varfurile unui triunghi, utilizand datele din VBO
    //  Functia primeste 3 argumente:
    //  - arg1 = tipul primitivei desenate,
    //  - arg2 = indicele primului punct de desenat din buffer,
    //  - arg3 = numarul de puncte consecutive de desenat;

    glDrawArrays(GL_LINE_STRIP, 0, 4);  // 4
    glDrawArrays(GL_POINTS, 4, 1);
    glDrawArrays(GL_LINES, 5, 2);   // 1
    glDrawArrays(GL_LINE_LOOP, 7, 4);   // 0
    glDrawArrays(GL_POINTS, 11, 1);
    glDrawArrays(GL_LINE_STRIP, 12, 6);  // 2
    glDrawArrays(GL_LINE_STRIP, 18, 7);  // 3


    glFlush();                          //  Asigura rularea tuturor comenzilor OpenGL apelate anterior;
}
```