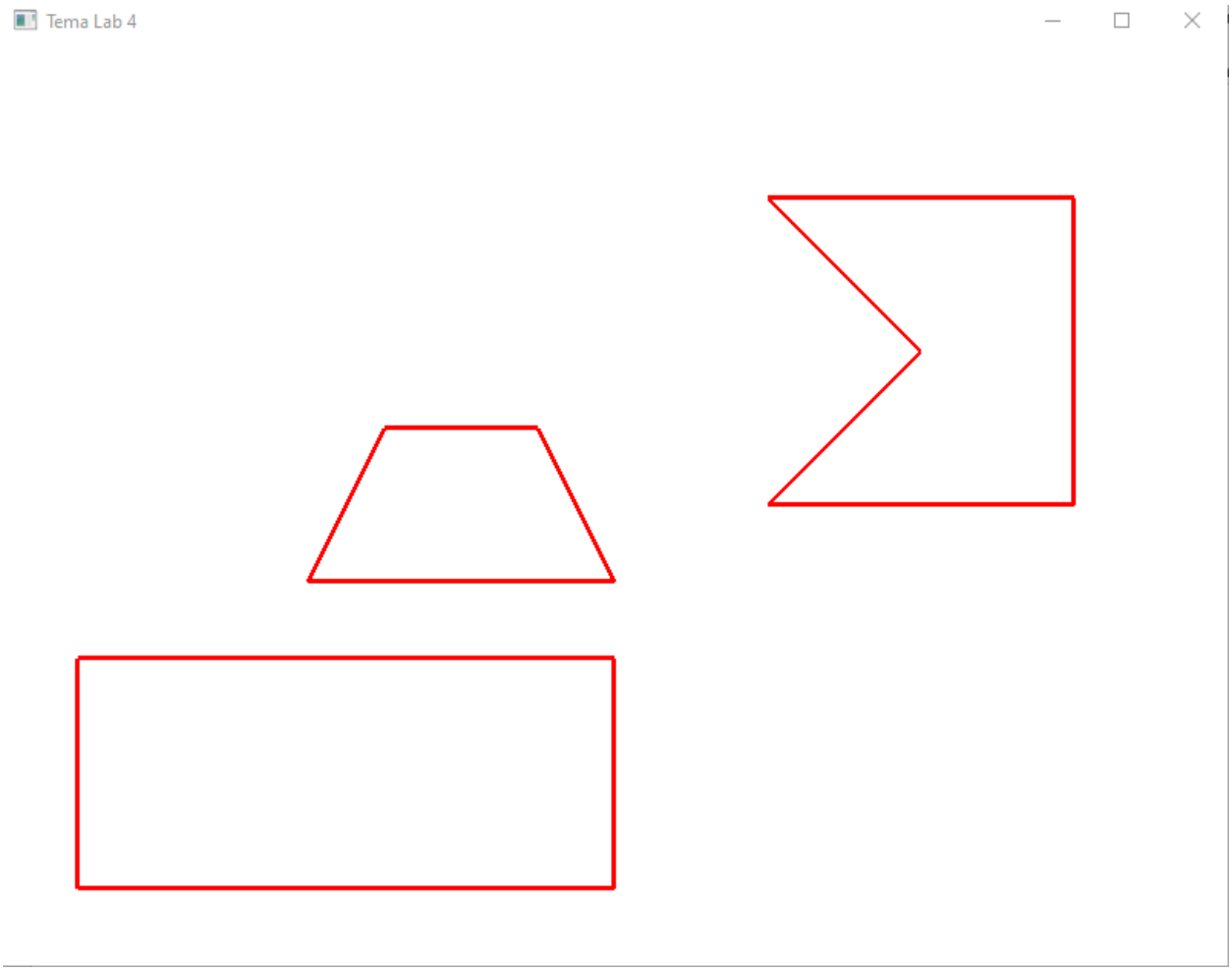
Am pornit de la codul sursa 04_03_transformari_glm.cpp.

1. Am aplicat o translatie de (400, 300), urmand ca apoi sa desenez figurile cerute.

```
void Initialize(void)
{

   resizeMatrix= glm::scale(glm::mat4(1.0f), glm::vec3(1.f/width, 1.f/height, 1.0));
   matrTransl=glm::translate(glm::mat4(1.0f), glm::vec3(-400.f, -300.f, 0.0));
```

```
void RenderFunction(void)
{
    glm::mat4 view;
    view = glm::lookAt(glm::vec3(0.0f, 0.0f, 5.f),
           glm::vec3(0.0f, 0.0f, -20.0f),
           glm::vec3(0.0f, 1.0f, 0.0f));

  glClear(GL_COLOR_BUFFER_BIT);
  myMatrix = resizeMatrix * matrTransl;
  // displayMatrix( );

    // matricea de redimensionare (pentru elementele "fixe")

  myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
  glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,  &myMatrix[0][0]);
    // desenare puncte din colturi si axe
  glLineWidth(3.0);
  //glDrawArrays(GL_POLYGON, 13, 4);

  glDrawArrays(GL_LINE_LOOP, 0, 4);

  glDrawArrays(GL_LINE_LOOP, 4, 4);

  glDrawArrays(GL_LINE_LOOP, 8, 5);
```

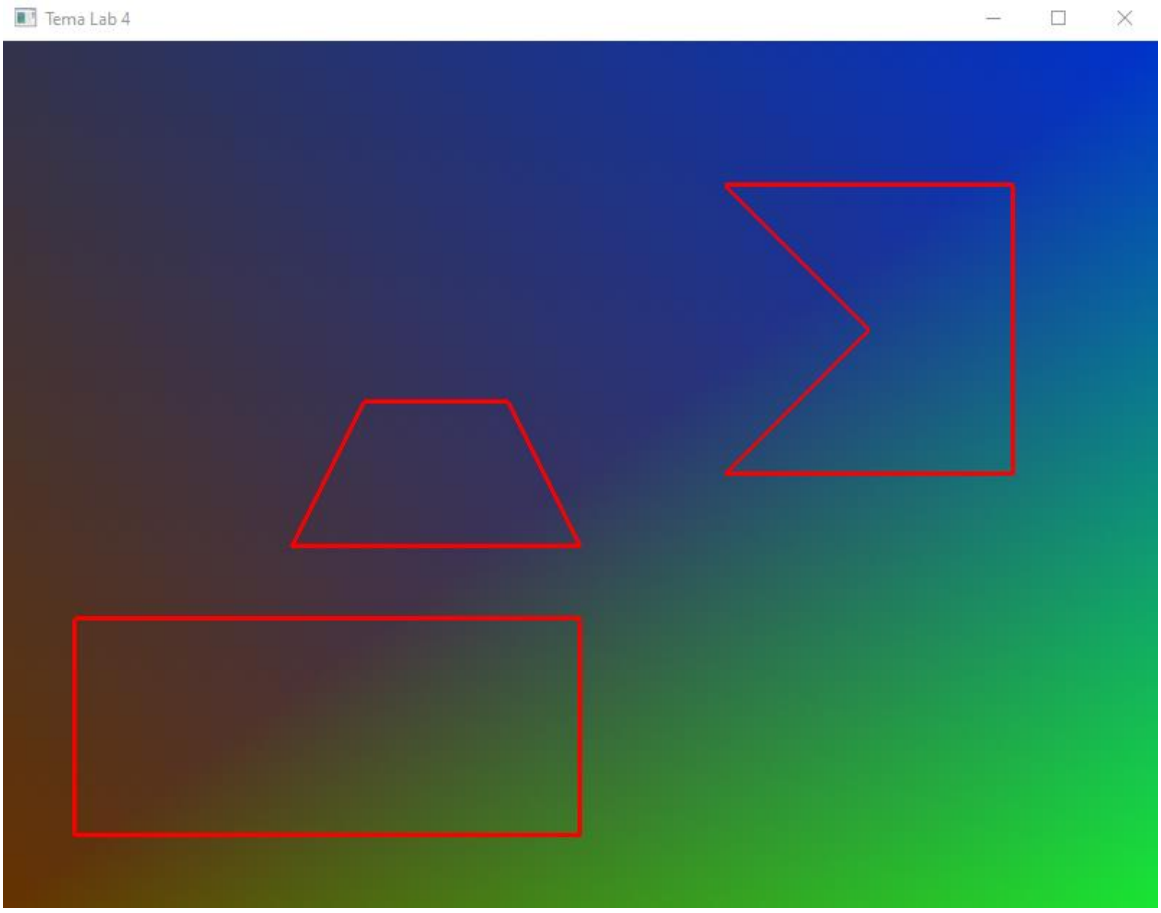
```
GLfloat Vertices[] = {
    // varfuri pentru dreptunghiul D
   50.0f,  50.0f, 0.0f, 1.0f,
   400.0f, 50.0f, 0.0f, 1.0f,
   400.0f,  200.0f, 0.0f, 1.0f,
   50.0f,  200.0f, 0.0f, 1.0f,
    // varfuri pentru poligonul P1 (trapez)
   200.0f,  250.0f, 0.0f, 1.0f,
   250.0f,  350.0f, 0.0f, 1.0f,
   350.0f,  350.0f, 0.0f, 1.0f,
   400.0f, 250.0f, 0.0f, 1.0f,
    // varfuri pentru poligonul P2
    500.0f,  300.0f, 0.0f, 1.0f,
    600.0f,  400.0f, 0.0f, 1.0f,
    500.0f,  500.0f, 0.0f, 1.0f,
    700.0f,  500.0f, 0.0f, 1.0f,
    700.0f,  300.0f, 0.0f, 1.0f,
    // varfuri pentru dreptunghiul de fundal
    0.0f,  0.0f, 0.0f, 1.0f,
    0.0f,  600.0f, 0.0f, 1.0f,
    800.0f,  600.0f, 0.0f, 1.0f,
    800.0f, 0.0f, 0.0f, 1.0f,
    // punct de rotatie
    450.0f,  325.0f, 0.0f, 1.0f,
};



// culorile varfurilor din colturi
GLfloat Colors[] = {
   // varfuri pentru dreptunghiul D
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  // varfuri pentru poligonul P1 (trapez)
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  // varfuri pentru poligonul P2
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f, 1.0f,
```



2. Am creat un dreptunghi cu varfurile in colturile ferestrei si l-am desenat inaintea celorlalte figuri.

```
GLfloat Colors[] = {
  // varfuri pentru dreptunghiul D
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 // varfuri pentru poligonul P1 (trapez)
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 // varfuri pentru poligonul P2
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 1.0f, 0.0f, 0.0f, 1.0f,
 // varfuri pentru dreptunghiul de fundal
 0.4f, 0.2f, 0.0f, 1.0f,
 0.2f, 0.2f, 0.3f, 1.0f,
 0.0f, 0.2f, 0.8f, 1.0f,
 0.1f, 0.9f, 0.2f, 1.0f,
};
```

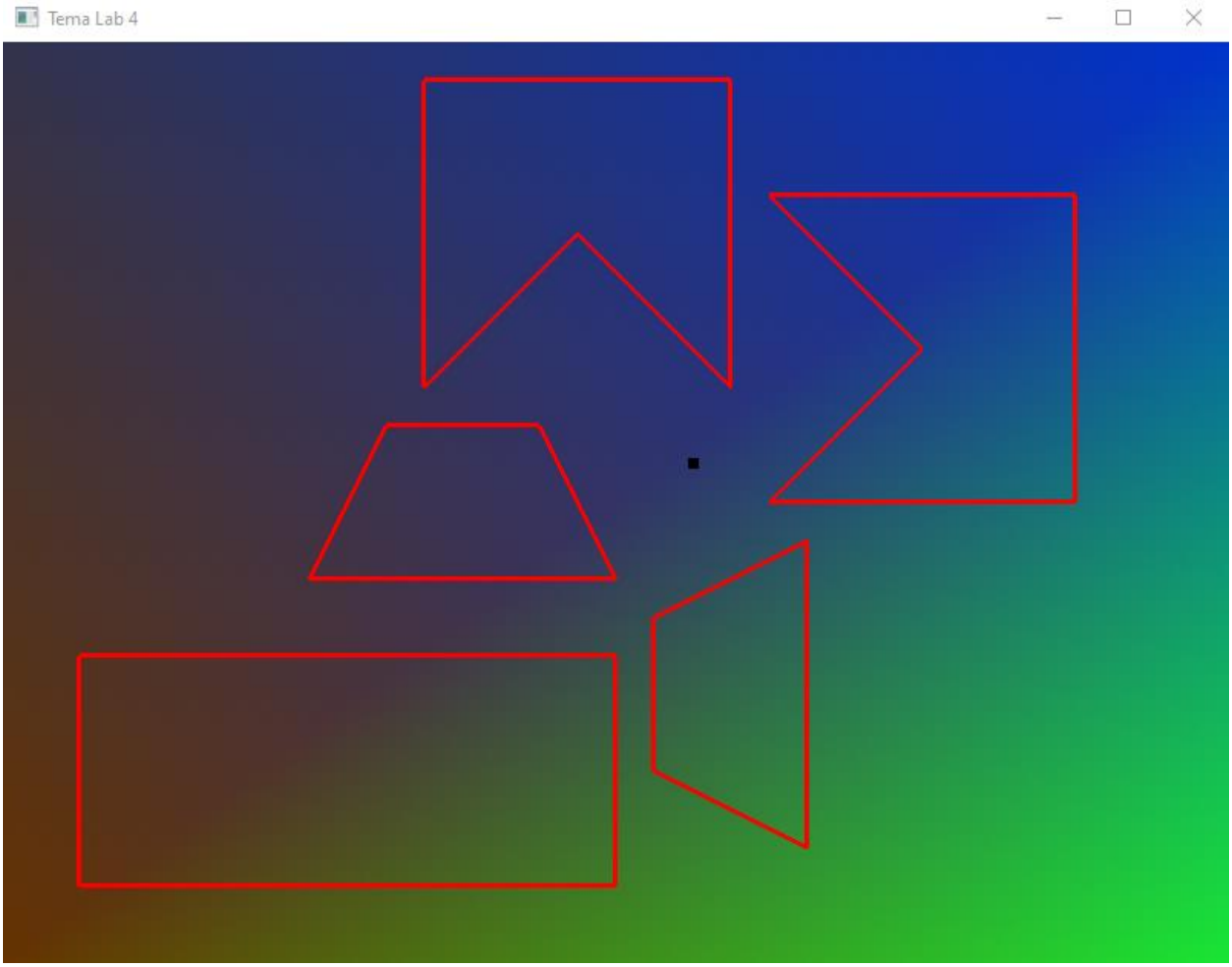
```
void RenderFunction(void)
{
    glm::mat4 view;
    view = glm::lookAt(glm::vec3(0.0f, 0.0f, 5.f),
           glm::vec3(0.0f, 0.0f, -20.0f),
           glm::vec3(0.0f, 1.0f, 0.0f));

  glClear(GL_COLOR_BUFFER_BIT);
  myMatrix = resizeMatrix * matrTransl;
  // displayMatrix( );

    // matricea de redimensionare (pentru elementele "fixe")

  myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
  glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,  &myMatrix[0][0]);
    // desenare puncte din colturi si axe
  glLineWidth(3.0);
  glDrawArrays(GL_POLYGON, 13, 4);

  glDrawArrays(GL_LINE_LOOP, 0, 4);

  glDrawArrays(GL_LINE_LOOP, 4, 4);

  glDrawArrays(GL_LINE_LOOP, 8, 5);
```



3. Am desenat punctul dintre cele doua poligoane.
   Am efectuat o rotatie de 90 de grade folosind metoda prezentata la laborator.

```
void Initialize(void)
{

    resizeMatrix= glm::scale(glm::mat4(1.0f), glm::vec3(1.f/width, 1.f/height, 1.0));
    matrTransl=glm::translate(glm::mat4(1.0f), glm::vec3(-400.f, -300.f, 0.0));

    matrRot = glm::rotate(glm::mat4(1.0f), PI/2, glm::vec3(0.0, 0.0, 1.0));
    matrTransl1 = glm::translate(glm::mat4(1.0f), glm::vec3(-450.0f, -325.0f, 0.0f));
    matrTransl2 = glm::translate(glm::mat4(1.0f), glm::vec3(450.0f, 325.0f, 0.0f));
```

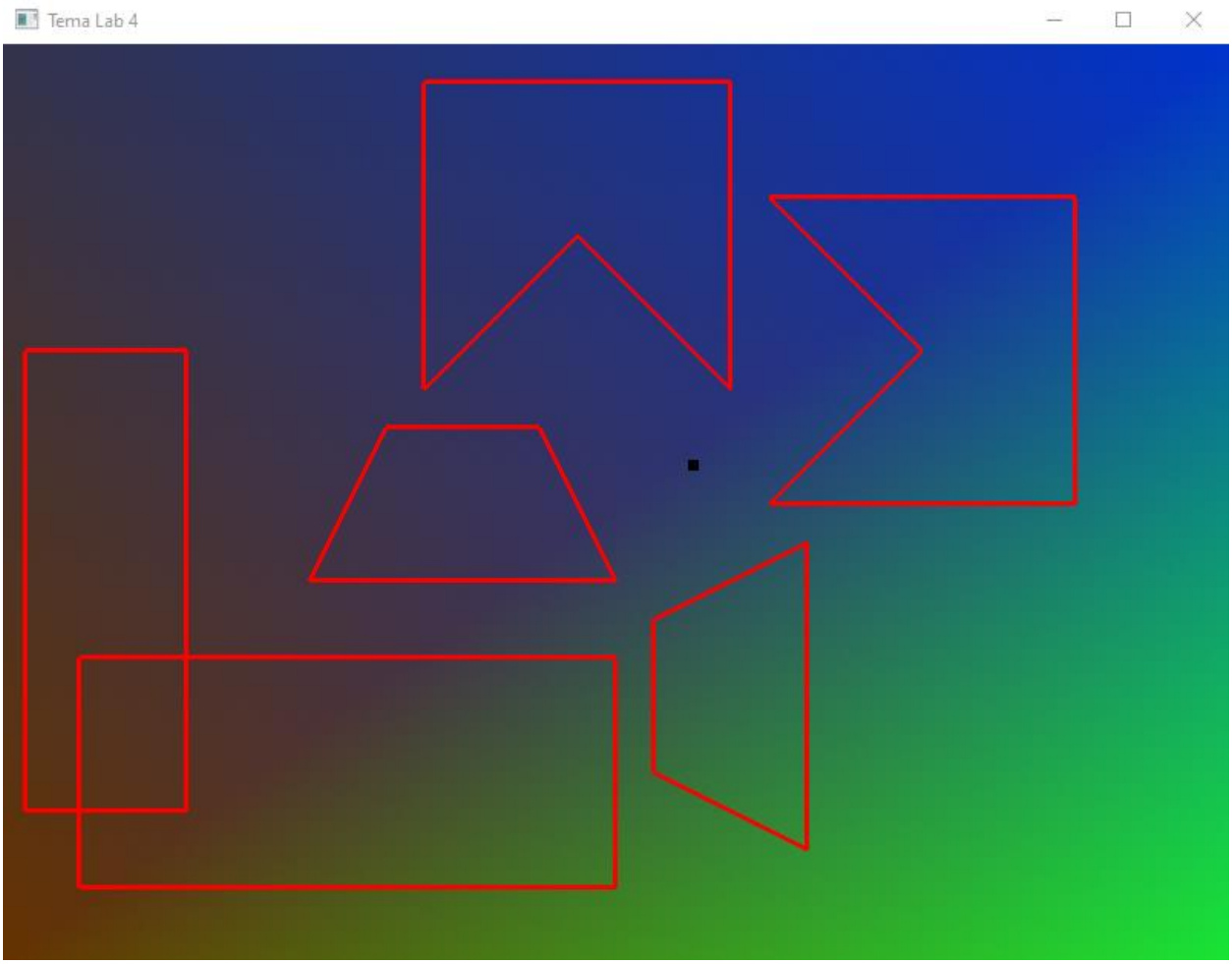
```
void RenderFunction(void)
{
   glm::mat4 view;
   view = glm::lookAt(glm::vec3(0.0f, 0.0f, 5.f),
          glm::vec3(0.0f, 0.0f, -20.0f),
          glm::vec3(0.0f, 1.0f, 0.0f));

  glClear(GL_COLOR_BUFFER_BIT);
  myMatrix = resizeMatrix * matrTransl;
  // displayMatrix( );

    // matricea de redimensionare (pentru elementele "fixe")

  myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
  glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,  &myMatrix[0][0]);
    // desenare puncte din colturi si axe
  glLineWidth(3.0);
  glDrawArrays(GL_POLYGON, 13, 4);

  glDrawArrays(GL_LINE_LOOP, 0, 4);

  glDrawArrays(GL_LINE_LOOP, 4, 4);

  glDrawArrays(GL_LINE_LOOP, 8, 5);

  glPointSize(7.0);
  glDrawArrays(GL_POINTS, 17, 1);


  myMatrix = myMatrix * matrTransl2 * matrRot * matrTransl1;
  myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
  glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);


  glDrawArrays(GL_LINE_LOOP, 4, 4);

  glDrawArrays(GL_LINE_LOOP, 8, 5);
```



4. Am scalat dreptunghiul folosind vectorul (0.3, 2.0, 1.0)

```
void Initialize(void)
{

    resizeMatrix= glm::scale(glm::mat4(1.0f), glm::vec3(1.f/width, 1.f/height, 1.0));
    matrTransl=glm::translate(glm::mat4(1.0f), glm::vec3(-400.f, -300.f, 0.0));

    matrRot = glm::rotate(glm::mat4(1.0f), PI/2, glm::vec3(0.0, 0.0, 1.0));
    matrTransl1 = glm::translate(glm::mat4(1.0f), glm::vec3(-450.0f, -325.0f, 0.0f));
    matrTransl2 = glm::translate(glm::mat4(1.0f), glm::vec3(450.0f, 325.0f, 0.0f));

    matrScale = glm::scale(glm::mat4(1.0f), glm::vec3(0.3f, 2.0f, 1.0));
```

```
void RenderFunction(void)
{
   glm::mat4 view;
   view = glm::lookAt(glm::vec3(0.0f, 0.0f, 5.f),
          glm::vec3(0.0f, 0.0f, -20.0f),
          glm::vec3(0.0f, 1.0f, 0.0f));

  glClear(GL_COLOR_BUFFER_BIT);
  myMatrix = resizeMatrix * matrTransl;
  // displayMatrix( );

    // matricea de redimensionare (pentru elementele "fixe")

  myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
  glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,  &myMatrix[0][0]);
    // desenare puncte din colturi si axe
  glLineWidth(3.0);
  glDrawArrays(GL_POLYGON, 13, 4);

  glDrawArrays(GL_LINE_LOOP, 0, 4);

  glDrawArrays(GL_LINE_LOOP, 4, 4);

  glDrawArrays(GL_LINE_LOOP, 8, 5);



  glPointSize(7.0);
  glDrawArrays(GL_POINTS, 17, 1);

  myMatrix = myMatrix * matrTransl2 * matrRot * matrTransl1;
  myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
  glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);


  glDrawArrays(GL_LINE_LOOP, 4, 4);

  glDrawArrays(GL_LINE_LOOP, 8, 5);



  myMatrix = resizeMatrix * matrTransl;
  myMatrix = myMatrix * matrScale;
  myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
  glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);

  glDrawArrays(GL_LINE_LOOP, 0, 4);
```


Tema Lab 4