

Universitatea din Oradea
Facultatea de Inginerie Electrică și Tehnologia Informației
Specializarea: Calculatoare



Proiect EGC – Geometria în spațiu

Tema 3

Coordonator:

dr.ing. Pater Alexandrina Mirela

Studenti:

Fărcuța Andrei Marian
Ienciu David Ștefan

Cuprins

Tema proiectului	3
I. CODIFICAREA IN PROCESSING	3
1.1. Funcția void setup().....	3
1.2. Funcția void draw()	3
1.3. Subprogramul drawCylinder()	5
II. Rezultatul implementării grafice	6
Bibliografie.....	8

Tema proiectului

În acest proiect ne propunem să implementăm în programul processing 5 corpuri geometrice diferite în format 3D. Formele a căror implementare nu este permisă sunt: piramidă triunghiulară, cub și paralelipiped.

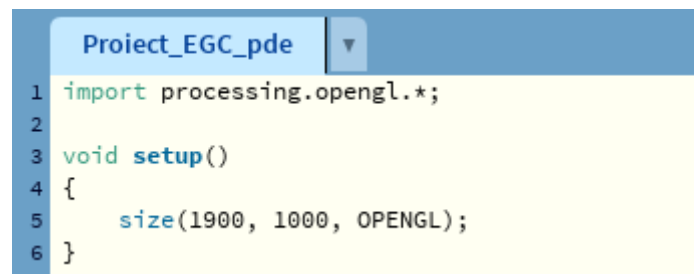
Vom implementa operațiuni de rotație pentru fiecare corp. De asemenea, vom folosi instrucțiunea `lights()` pentru a scoate în evidență fețele formelor. Pentru a schimba culoarea fețelor utilizatorul va avea posibilitatea de a schimba valorile din instrucțiunea `fill()`, iar pentru a schimba culoarea liniilor acesta va interacționa cu valorile din `stroke()`. Pentru schimbarea background-ului se va folosi instrucțiunea `background()`.

Utilizatorul va avea posibilitatea de a modifica formele prin alegerea valorilor din `drawCylinder(int, float, float, float)`. Formele pe care le-am ales pentru implementare sunt sfera, cilindrul, octagonul, trunchiul de piramidă hexagonală și conul.

I. CODIFICAREA ÎN PROCESSING

1.1. Funcția void setup()

În primul rând, importăm din librăriile processing OpenGL pentru a putea crea forme complexe 3D în program. În funcția `void setup()` setăm dimensiunile ferestrei unde vom implementa formele geometrice.

A screenshot of a code editor window titled "Proiect_EGC_pde". The code is as follows:

```
1 import processing.opengl.*;
2
3 void setup()
4 {
5     size(1900, 1000, OPENGL);
6 }
```

Figura 1.1. Librării utilizate

1.2. Funcția void draw()

Funcția `draw()` execută continuu liniile de cod conținute în blocul său până când programul este oprit. În funcția `draw()` alegem background-ul, culorile formelor, implementăm operațiunile de rotație pentru fiecare corp și apelăm funcțiile/metodele necesare pentru crearea figurilor geometrice.

Vom folosi în program metoda `pushMatrix()` pentru a salva sistemul de coordonate curent în stivă, iar `popMatrix()` pentru a restabili sistemul de coordonate anterior. Metoda `translate()` este folosită pentru a alege poziția fiecărei figuri geometrice în fereastră, aceasta fiind apelată de 5 ori pentru 5 forme.

Pentru crearea sferei am folosit metoda `sphere()`, iar pentru crearea celorlalte forme am creat un subprogram ușor de utilizat.

```
8 void draw()
9 {
10     background(0);
11     lights();
12     stroke(255);
13     fill(255, 0, 0);
14     pushMatrix();
15     translate( width/3,height/4 );
16     rotateX( PI/6 );
17     rotateY( radians( frameCount ) );
18     rotateZ( radians( frameCount ) );
19     drawCylinder( 30, 100, 100, 200 );
20     popMatrix();
21
22     pushMatrix();
23     translate( width/1.5,height/4 );
24     rotateX( PI/4 );
25     rotateY( radians( frameCount ) );
26     rotateZ( radians( frameCount ) );
27     drawCylinder( 50, 100, 0, 150 );
28     popMatrix();
29
30     pushMatrix();
31     translate(width/2,height/2);
32     rotateX( PI/4 );
33     rotateY( radians( frameCount ) );
34     rotateZ( radians( frameCount ) );
35     sphere(100);
36     popMatrix();
37
38     pushMatrix();
39     translate( width/3,height/1.3 );
40     rotateX( PI/2 );
41     rotateY( radians( frameCount ) );
42     rotateZ( radians( frameCount ) );
43     drawCylinder( 8, 100,0, 100 );
44     popMatrix();
45
46     pushMatrix();
47     translate( width/1.5,height/1.3 );
48     rotateX( PI/12 );
49     rotateY( radians( frameCount ) );
50     rotateZ( radians( frameCount ) );
51     drawCylinder( 6, 100, 160, 100 );
52     popMatrix();
53 }
```

Figura 1.2. Codificarea funcției `void draw()`

1.3. Subprogramul drawCylinder()

În subprogramul drawCylinder() sunt solicitate valori pentru sides, r1, r2 și h. Subprogramul are ca scop de bază construirea de cilindrii, acesta solicitând dimensiunile specifice. Prin sides înțelegem numărul de fețe laterale al figurii geometrice, r1 influențează mărimea feței inferioare, r2 mărimea feței superioare (a așa zisului cilindru), iar h reprezintă înălțimea forme geometrice.

Construirea forme este împărțită în 3 etape, delimitate de beginShape() și endShape(CLOSE). În prima etapă corespunde construirii feței inferioare, a doua etapă construirii feței superioare, iar a treia etapă constă în construirea fețelor laterale, al căror număr este dictat de variabila sides.

```
55 void drawCylinder( int sides, float r1, float r2, float h)
56 {
57     float angle = 360 / sides;
58     float halfHeight = h / 2;
59
60     // draw top of the tube
61     beginShape();
62     for (int i = 0; i < sides; i++) {
63         float x = cos( radians( i * angle ) ) * r1;
64         float y = sin( radians( i * angle ) ) * r1;
65         vertex( x, y, -halfHeight);
66     }
67     endShape(CLOSE);
68
69     // draw bottom of the tube
70     beginShape();
71     for (int i = 0; i < sides; i++) {
72         float x = cos( radians( i * angle ) ) * r2;
73         float y = sin( radians( i * angle ) ) * r2;
74         vertex( x, y, halfHeight);
75     }
76     endShape(CLOSE);
77
78     // draw sides
79     beginShape(TRIANGLE_STRIP);
80     for (int i = 0; i < sides + 1; i++) {
81         float x1 = cos( radians( i * angle ) ) * r1;
82         float y1 = sin( radians( i * angle ) ) * r1;
83         float x2 = cos( radians( i * angle ) ) * r2;
84         float y2 = sin( radians( i * angle ) ) * r2;
85         vertex( x1, y1, -halfHeight);
86         vertex( x2, y2, halfHeight);
87     }
88     endShape(CLOSE);
89 }
90 }
```

Figura 1.3. Metoda de creare a obiectelor

II. Rezultatul implementării grafice

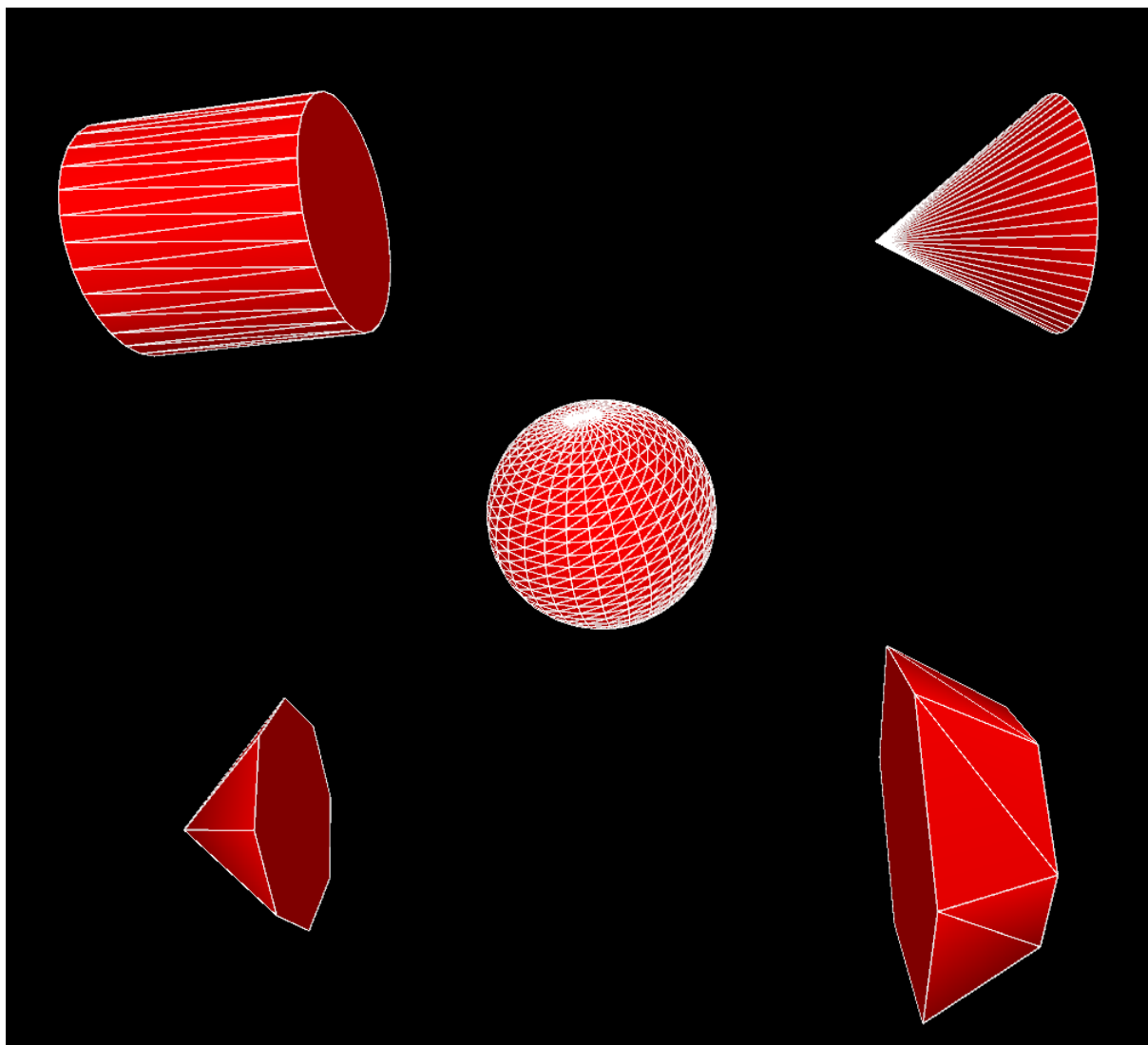


Figura 2.1. Implementarea grafică conform codului sursă

Folosind funcția `noFill()` putem observa cel mai bine fețele figurilor geometrice.

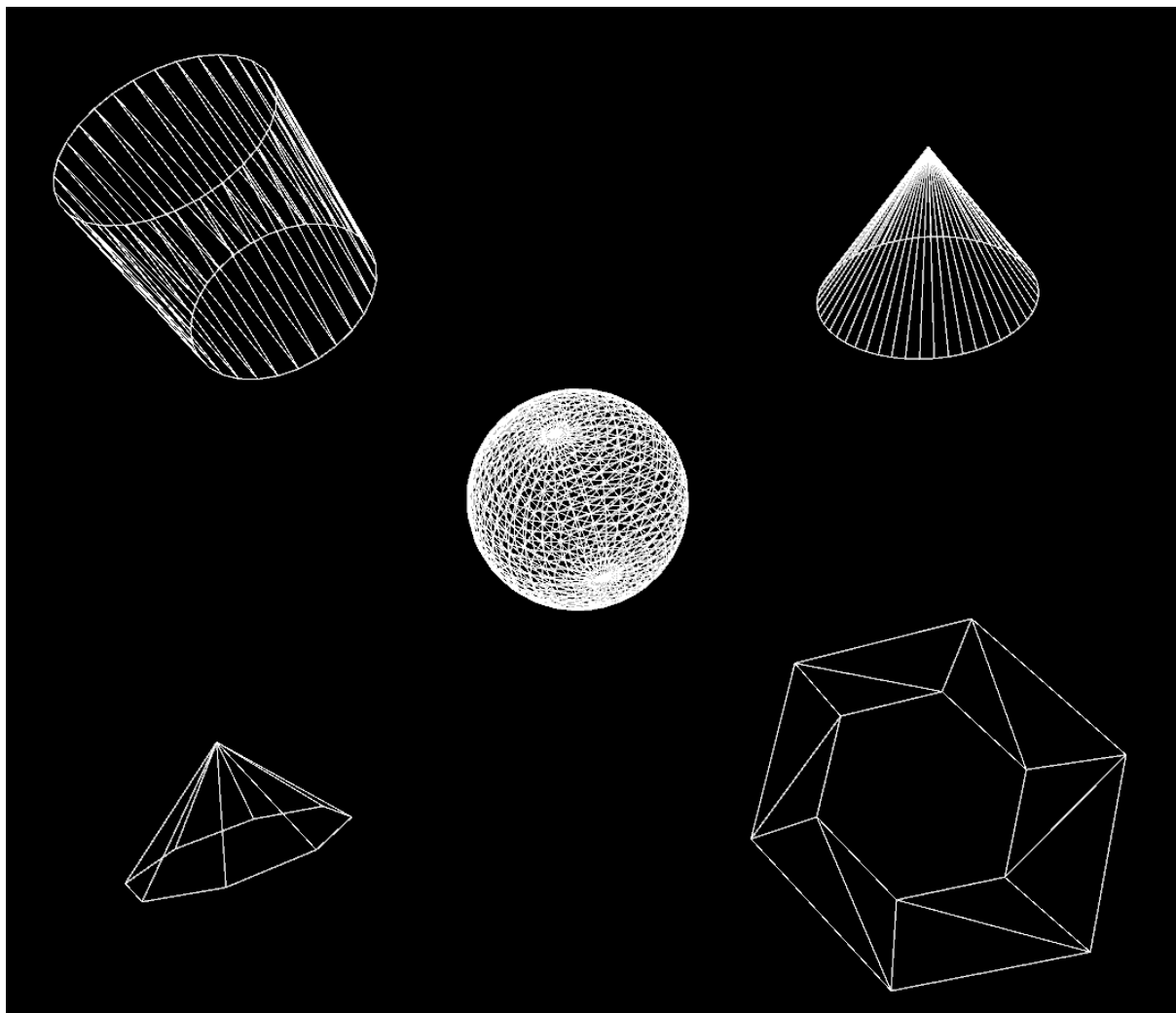


Figura 2.2. Implementarea grafică folosind `noFill()`

Bibliografie

- [1] https://processing.org/reference/sphere_.html
- [2] <https://processing.org/tutorials/p3d>
- [3] <https://vormplus.be/full-articles/drawing-3d-shapes-with-processing>
- [4] <https://forum.processing.org/one/topic/draw-a-cone-cylinder-in-p3d.html>