UNIVERSITATEA TEHNICĂ A MOLDOVEI

Rotaru Lilia, Oșovschi Mariana,
Cărbune Viorel, Ursu Adriana

GRAFICA PE CALCULATOR

ÎNDRUMAR METODIC PENTRU LUCRĂRILE DE LABORATOR

Chișinău
2025

VI. MODELAREA PROCESELOR 3D DINAMICE

Modelarea proceselor 3D dinamice, pe lângă scopuri pur științifice, poate avea și valoare aplicativă.

Biblioteca grafică p5.js poate fi utilizată pentru modelarea fizică a proceselor dinamice. Aceasta permite modelarea sistemelor mecanice (în cadrul legilor mecanicii teoretice). Cu ajutorul p5.js pot fi simulate mișcări de translație și rotație în trei planuri.

În exemplul prezentat în figura 6.1 este simulată ciocnirea a două sfere cu masele m_1 și m_2, care până la ciocnire au vitezele deplasării liniare a centrelor notate cu v_1 și v_2. După ciocnire, sferele se vor deplasa cu vitezele v'_1 și v'_2. Sferele se rostogolesc pe o suprafață rigidă, adică execută o mișcare de rotație specificată de unghiuri: ”omega$_1$” și ”omega$_2$”.

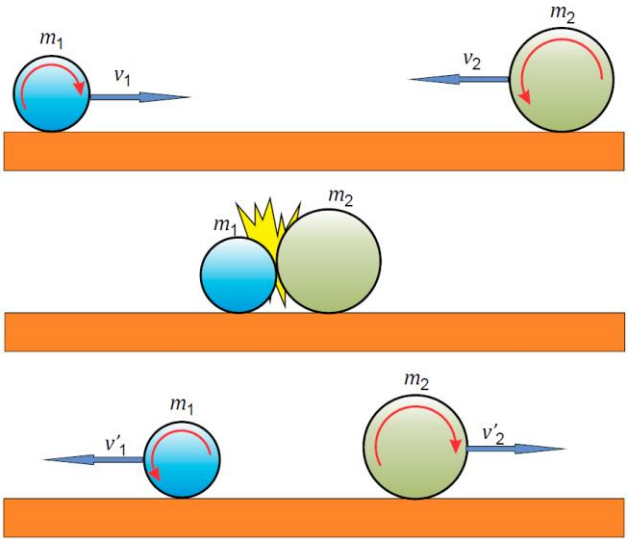


Figura 6.1. Ciocnirea a două sfere

Utilizând legea conservării energiei mecanice din formula (6.1):

$$\frac{m_1 v_1^2}{2} + \frac{m_2 v_2^2}{2} = \frac{m_1 v'^2_1}{2} + \frac{m_2 v'^2_2}{2} \quad (6.1)$$

și legea conservării impulsului din formula (6.2):

$$m_1 v_1 + m_2 v_2 = m_1 v'_1 + m_2 v'_2, \quad (6.2)$$

pot fi determinate relațiile dintre viteze până și după ciocnirea sferelor (formulele 6.3 și 6.4):

$$v'_1 = -v_1 + 2\frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}, \quad (6.3)$$

$$v'_2 = -v_2 + 2\frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}. \quad (6.4)$$

Unghiul de rotație poate fi exprimat cu ajutorul formulelor 6.5 și 6.6.

$$\text{omega}_1 = \frac{r_1}{v_1}; \quad (6.5)$$

$$\text{omega}_2 = \frac{r_2}{v_2}. \quad (6.6)$$

Modelul acestui experiment poate fi descries de următorul cod:

Listingul programului:

```
let x1=-150;
let v1=5;
let m1=50;
let omega1=v1/m1;
let a1=0;
let x2=150;
let v2=-15;
let m2=20;
let omega2=v2/m2;
let a2=0;
function setup()
{  createCanvas(500, 500, WEBGL);
  strokeWeight(0.2);
  frameRate(24); }
function draw()
```

```
{   background(200);
  orbitControl();
  pointLight(255, 255, 255, 0, 0, 1000);
  push();
  box(500,100,100);
  pop();
  push();
  translate(x1, -100/2-m1, 0);
  rotateZ(a1);
  sphere(m1);
  pop();
  push();
  translate(x2, -100/2-m2, 0);
  rotateZ(a2);
  sphere(m2);
  pop();
  if(x1+m1>x2-m2)
    { v1=2*(m1*v1+m2*v2)/(m1+m1)-v1;
      v2=2*(m2*v2+m1*v1)/(m1+m1)-v2;
      x1=x1+v1;
      x2=x2+v2;
      omega1=v1/m1;
      omega2=v2/m2;     }
      x1=x1+v1;
      x2=x2+v2;
      a1=a1+omega1;
      a2=a2+omega2;  }
```

Descrierea programului:

let – creează și denumește o variabilă nouă. O variabilă este un container pentru a memora a valoare.

Variabilele care sunt declarate cu **let** vor avea domeniul de aplicare bloc. Aceasta înseamnă că variabila există numai în blocul în care este creată.

În programul de mai sus, în calitate de variabile considerăm:

– x1 și x2 sunt pozițiile inițiale ale sferelor;

– v1 și v2 sunt vitezele liniare inițiale ale centrelor sferelor;

– masa bilelor m1 și m2 care exprimă și razele sferelor;

– omega1=v1/m1 și omega2=v2/m2 sunt unghiurile de rotație a sferelor;

– a1 = 0 și a2 = 0 sunt unghiurile inițiale de rotație a sferelor.

Corpul pe care se rostogolesc sferele este construit cu ajutorul funcției **box(500,100,100),** fiind un obiect static.

Sferele sunt desenate cu ajutorul funcției **sphere()**, asupra lor sunt aplicate mișcarea de translație cu ajutorul funcției **translate()** și de rotație cu ajutorul funcției **rotateZ().**

Modelele corpurilor sunt apelate fiecare în blocul său, fiind delimitate cu **push()**, **pop()**.

Ultima parte a programului include formulele de calcul pentru vitezele sferelor, pozițiilor acestora și al unghiului de rotație.

Rezultatul execuției programului este dat în figura 6.2.

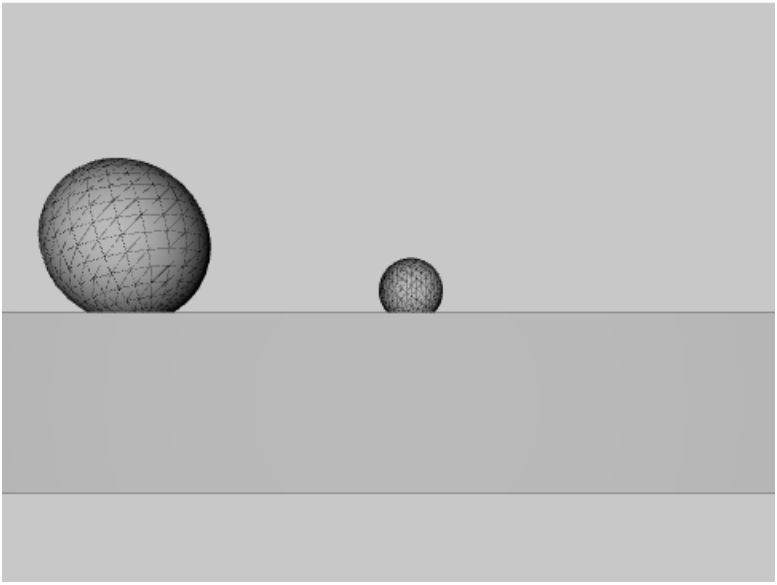


Figura 6.2. Modelul simulării ciocnirii a 2 sfere în p5.js

Pentru realizarea modelului se utilizează primitivele 3D din biblioteca p5.js.

6.1 Crearea primitivelor grafice 3D simple

Pentru crearea corpurilor în p5.js sunt utilizate primitivele grafice 3D simple cum ar fi:

- `plane()`
- `box()`
- `sphere()`
- `cylinder()`
- `cone()`
- `ellipsoid()`
- `torus()`

Funcția plane(): desenează un plan cu o lățime și o înălțime dată

Sintaxa:

```
plane([width],[height], [detailX], [detailY]);
```

unde:

[width]: lațimea (obțional)

[height]: înnățimea (obțional))

detailX: numărul de triunghiuri pe axa x (opțional)

detailY: numărul de triunghiuri pe axa y (opțional)

Funcția box(): desenează un paralelepiped cu lățimea, înălțimea și adâncimea specificată

Sintaxa:

```
box([width],[Height],[depth],[detailX],
[detailY]);
```

unde:

[width]: lațimea figurii (obțional)

[height]: înnățimea figurii (obțional))

[depth]: adîncimea figurii (obțional)

detailX: numărul de triunghiuri pe axa x (opțional)

detailY: numărul de triunghiuri pe axa y (opțional)

Funcția sphere(): Desenează o sferă cu raza specificată. DetailX și detailY determină numărul de segmente în dimensiunea x și dimensiunea y a sferei. Mai multe segmente fac sfera mai netedă. Valoarea maximă recomandată este 24. Utilizarea unei valori mai mari de 24 poate face lucrul browserului mai lent.

Sintaxa:

```
sphere([radius], [detailX], [detailY]);
```

unde:

radius: raza cerdcului (opțional)

detailX: numărul de segmente pe axa x (opțional)

detailY: numărul de segmente pe axa y (opțional)

Funcția cylinder(): desenează un cilindru cu raza și înălțimea specificată. DetailX și detailY determină numărul de segmente pe axa x și axa y a cilindrului. Mai multe segmente fac ca cilindrul să pară mai neted. Valoarea maximă recomandată pentru detailX este 24. Utilizarea unei valori mai mari de 24 poate încetini browserul.

Sintaxa:

```
cylinder([radius], [height], [detailX], [detailY], [bottomCap], [topCap]);
```

unde:

radius: raza bazei cilindrului (opțional)

height: înnălțimea cilindrului (opțional)

detailX: numărul de segmente pe axa x, implicit 24 (opțional)

detailY: numărul de segmente pe axa y, implicit 1 (opțional)

bottomCap Boolean: să deseneze sau nu partea de jos a cilindrului (opțional)

topCap Boolean: să deseneze sau nu partea de sus a cilindrului (opțional)

Funcția cone(): desenează un con cu rază și înălțimea dată. DetailX și detailY determină numărul de segmente pe axa x și axa y a conului. Mai multe segmente fac conul să pară mai neted. Valoarea maximă recomandată pentru detailX este 24. Utilizarea unei valori mai mari de 24 poate încetini browserul.

Sintaxa:

```
cone([radius], [height], [detailX], [detailY], [cap]);
```

unde:

radius: raza bazei conului (opțional)

height: înnălțimea conului (opțional)

detailX: numărul de segmente pe axa x, implicit 24 (opțional)

detailY: numărul de segmente pe axa y, implicit 1 (opțional)

Cap Boolean: să deseneze sau nu partea baza cunului (opțional)

Funcția ellipsoid(): desenează un elipsoid. DetailX și detailY determină numărul segmente pe axa x și axa y a figurii. Mai multe segmente fac elipsoidul mai neted. Evitați numărul de detaliere mai mare de 150, deoarece acesta poate bloca browserul.

Sintaxa:

```
ellipsoid([radiusx], [radiusy], [radiusz], [detailX], [detailY]);
```

unde:

radiusx: x-raza elipsoidului (opțional)

radiusy: y- raza elipsoidului (opțional)

radiusz: z- raza elipsoidului (opțional)

detailX: numărul de segmente pe axa x, implicit este 24. Evitați numărul segmentelor mai mare de 150 poate bloca browserul. (Opțional)

detailY: numărul de segmente pe axa y, implicit este 16. Mai mare de 150 poate bloca browserul. (Opțional)

Funcția torus(): desenează un inel. DetalX și detalY determină numărul de segmente pe axa x și axa y a torului. Valorile maxime implicite pentru detailX și detailY sunt 24 și, respectiv, 16. Setarea lor la valori relativ mici, cum ar fi 4 și 6, vă permite să creați noi forme, altele decât un tor.

Sintaxa:

```
torus([radius], [tubeRadius], [detailX], [detailY]);
```

Parametrii:

radius: raza externă a figurii (Opțional)

tubeRadius: raza internă a figurii (Opțional)

detailX: numărul de segmente pe axa x, implicit este 24 (opțional).

detailY: numărul de segmente pe axa y, implicit este 16 (opțional).

Lucrarea de laborator nr. 6
Tema: MODELAREA PROCESELOR ÎN 3D

Obiectivele lucrării:

1. Familiarizarea cu procesele fizice utilizate pentru crearea scenelor grafice 3D.

2. Utilizarea primitivelor grafice – obținerea cunoștințelor practice privind utilizarea primitivelor grafice 3D pentru crearea scenei.

3. Crearea scenelor grafice 3D dinamice – dezvoltarea abilităților de creare și compunere a scenelor grafice 3D dinamice prin combinarea eficientă a transformărilor geometrice.

4. Înțelegerea și aplicarea conceptelor de procese fizice – utilizarea noțiunilor de viteză, putere, gravitaţie și altele utilizate pentru dirijarea dinamicii vizuale în grafica 3D.

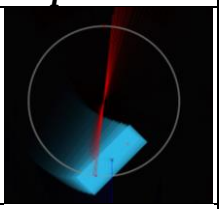
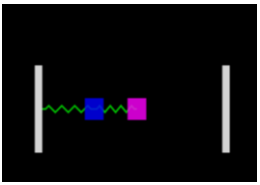
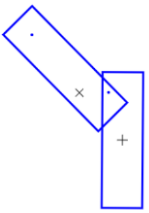
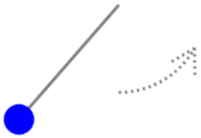
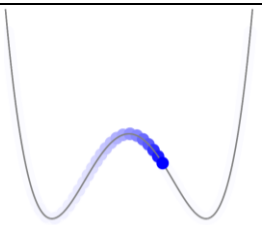
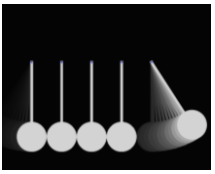
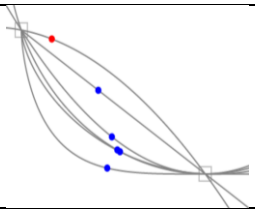
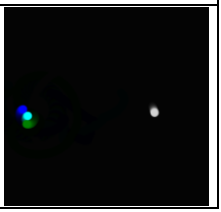
Numărul de ore necesare pentru realizare – 4 ore academice.

Scopul lucrării: obținerea cunoștințelor practice privind crearea scenelor 3D în modelarea proceselor 3D dinamice prin utilizarea funcțiilor standard de transformare geometrică, cum ar fi translația, scalarea și rotația din biblioteca p5.js.

Sarcina lucrării: elaborați un program care creează o scenă 3D în care să modelați un proces fizic, utilizând funcțiile standard de translație și rotație din biblioteca p5.js, conform variantei indicate în tabelul 6.1. Pentru crearea scenei pot fi utilizate obiecte grafice 3D existente în repozitorul 3D.

Simularea și descrierea procesului fizic bidimensional din sarcină pot fi consultate pe pagina din [6].

Tabelul 6.1. Variante pentru realizarea lucrării de laborator

Nr.	*Model*	*Reprezentare*	*Nr.*	*Model*	*Reprezentare*
1	Arc de extensie singular		6	Pendul rigid cu mișcare pe cerc	
2	Arc de extensie dublu		7	Pendul dublu rigid	
3	Pendul haotic		8	Coliziunea blocurilor rigide	
4	Mișcarea sferei pe traiectorie de tip cocoașă		9	Pendulul lui Newton	
5	Mișcarea sferei pe traiectorii		10	Atracția corpurilor	

Criterii de evaluare

1. **Corectitudinea codului** (20%) – verificarea corectitudinii codului, fără erori de sintaxă și funcționare.
2. **Utilizarea primitivelor grafice 3D** (20%) – evaluarea utilizării corecte și varietatea primitivelor grafice în conformitate cu cerințele lucrării.
3. **Respectarea instrucțiunilor și cerințelor** (10%) – verificarea corectitudinii cerințelor sarcinii cum ar fi anumite forme sau dimensiuni specifice ale primitivelor grafice și dinamica lor.
4. **Optimizarea codului** (10%) – evaluarea eficienței codului în utilizarea resurselor și evitarea codului redundant.
5. **Interactivitatea** (10%) – răspunsuri la mouse sau tastatură, se va evalua dacă aceste funcții sunt implementate corect.
6. **Estetica vizuală** (10%) – analiza atractivității vizuale a compoziției, cum ar fi armonia culorilor, proporțiile și echilibrul grafic.
7. **Respectarea termenului de predare** (10%) – evaluarea punctajului în funcție de punctualitate, dacă lucrarea a fost predată în termenul stabilit.
8. **Evaluarea cunoștințelor** (10%) – explicații privind procesul de realizare a lucrării, ceea ce poate include descrierea funcțiilor principale și a logicii utilizate.

Întrebări de verificare a cunoștințelor

1. Enumerați primitivele grafice 3D simple.
2. Enumerați funcțiile utilizate pentru transformările grafice.
3. Cum poate fi realizată modificarea atributelor de afişare a primitivelor grafice 3D?
4. Numiți formate de rotație și funcțiile corespunzătoare.
5. Ce primitive grafice pot fi utilizate pentru a crea forme de bază într-o scenă grafică 2D?
6. Cum poate fi realizată combinarea transformărilor grafice?
7. Ce rol are funcția push() și pop() în gestionarea transformărilor geometrice?
8. Cum se activează modul 3D în p5.js?

BIBLIOGRAFIE

1. L. McCarthy, C. Reas, and B. Fry, *Getting started with p5.js: making interaktive graphics in JavaScript and Processing*, First edition. in Make. San Francisco, CA: Maker Media, 2016.
2. E. Arslan, *Learn JavaScript with p5.js: coding for visual learners*. Place of publication not identified: Apress, 2018.
3. Referințele limbajului p5.js https://p5js.org/reference/
4. p5.js Overview https: //github.com/processing/p5.js/wiki/p5.js-overview
5. Cornel Marin, Modelarea sistemelor mecanice https://regielive.net/cursuri/mecanica/modelarea-sistemelor-mecanice -100377.html
6. Physics Simulations. Erik Neumann https://www.myphysicslab.com/
7. Proiectare 3D https://3dprint.capib.ro/proiectare-3d/
8. Cura Settings Decoded by Matt Jani, 2022 https://all3dp.com/1/cura-tutorial-software-slicer-cura-3d/
9. Online 3D printing service https://www.hubs.com/3d-printing/
10. https://openlab.bmcc.cuny.edu/makerspace/drawing-in-p5-js/

ANEXĂ

Modelul Raportului la lucrarea de laborator

Ministerul Educației și Cercetării
al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică
și Microelectronică

Raport la
lucrarea de laborator nr. 1
Disciplina: Grafica pe calculator

Tema: Sinteza imaginii 2D

Au efectuat: Nume Prenume
studentul gr: TI-241

A verificat: Nume Prenume
(profesorului)

Chișinău
2025

Scopul lucrării: dobândirea cunoștințelor practice privind crearea și manipularea scenelor grafice 2D statice, utilizând primitivele grafice oferite de biblioteca p5.js. Aplicarea cunoștințelor teoretice în sinteza imaginilor grafice, utilizarea eficientă a funcțiilor bibliotecii p5.js și crearea imaginii grafice 2D simple.

Sarcina lucrării: elaborați un program pentru sinteza unei scene 2D statice, utilizând cel puțin 6 primitive grafice diferite cum ar fi - *arc(), ellipse(), circle(), line(), point(), quad(), rect(), square(), triangle()*; primitivele trebuie să aibă diferite atribute, lucrarea trebuie semnată (numele, prenumele, grupa) în colțul din dreapta de jos a ecranului.

Varianta #

Program realizat în p5.js:

```
//Declarăm variabilele de sistem
var Width;
var Height;
var CurrentY;
function setup() {
  Width = 400;
  Height = 734;
  createCanvas(Width, Height);
}
function draw() {
  background(220);
  CurrentY = Height - 20;
 //Realizăm baza (1 parte)
  stroke(6, 21, 131);
  for (let i = 0; i < 20; i++) {
    line(0 + 20, CurrentY, Width - 20,
CurrentY);
    CurrentY--;
  }
 // Realizăm baza (2 parte)
```

```
       stroke(15, 23, 71);
      for (let i = 0; i < 20; i++) {
        line(0 + 20 + i, CurrentY, Width - 20 - i,
CurrentY);
        CurrentY--;
      }
      CurrentY += 10;
      stroke(6, 21, 121);
      for (let i = 0; i < 550; i++) {
        line(0 + 40, CurrentY, Width - 40,
CurrentY);
        CurrentY--;
      }
      fill(6, 21, 121);
      stroke(15, 21, 71);
      rect(40, CurrentY, 40, Height - 190);
      rect(80, CurrentY, 3, Height - 190);
      rect(83, CurrentY, 6, Height - 190);
      rect(Width - 40, CurrentY, -40, Height -
190);
      rect(Width - 80, CurrentY, -3, Height -
190);
      rect(Width - 83, CurrentY, -6, Height -
190);
      for (let i = 0; i < 4; i++) {
        for (let j = 0; j < 2; j++) {
          stroke(129, 153, 193);
          line(110 + j * 100, Height - 80 - i *
130, 110 + j * 100, Height - 180 - i * 130);
          line(110 + j * 100, Height - 80 - i *
130, 190 + j * 100, Height - 80 - i * 130);
          stroke(10, 14, 45);
          line(110 + j * 100, Height - 180 - i *
130, 190 + j * 100, Height - 180 - i * 130);
```

```
          line(190 + j * 100, Height - 180 - i *
130, 190 + j * 100, Height - 80 - i * 130);
        }
      }

      stroke(10, 14, 45);
      rect(200, CurrentY, -3, Height - 190);
      stroke(16, 31, 138);
      rect(203, CurrentY, -3, Height - 190);
      stroke(49, 65, 157);
      rect(205, CurrentY, -1, Height - 190);
      fill(240, 240, 240);
      ellipse(210, 350, 8, 30);
      fill(147, 127, 68);
      circle(210, 410, 10);
      stroke(10, 14, 45);
      line(110, Height - 80 - 2 * 130, 110, Height
- 180 - 2 * 130);
      line(110, Height - 80 - 2 * 130, 190, Height
- 80 - 2 * 130);
      line(110, Height - 180 - 2 * 130, 190,
Height - 180 - 2 * 130);
      line(190, Height - 180 - 2 * 130, 190,
Height - 80 - 2 * 130);
      fill(240, 240, 240);
      stroke(240, 240, 240);
      rect(120, Height - 430, 60, 80);
      fill(0, 0, 0);
      noStroke();
      textSize(5);
      text('POLICE TELEPHONE', 125, Height - 420);
      textSize(10);
      text('FREE', 135, Height - 405);
      textSize(5);
```

```
      text('FOR USE OR', 132, Height - 395);
      textSize(10);
      text('PUBLIC', 130, Height - 380);
      textSize(5);
      text('ADVICE & ASSIS', 128, Height - 370);
      textSize(7);
      text('PULL TO OPEN', 125, Height - 355);
      for(let j = 0; j < 2; j++) {
        stroke(129, 153, 193);
        fill(240, 240, 240);
        rect(113 + j * 100, Height-565, 75,93);
        stroke(18, 34, 129);
        line(138 + j * 100, Height-565, 138 + j *
100,Height-473);
        line(163 + j * 100, Height-565, 163 + j *
100,Height-473);
        line(113 + j * 100, Height-519, 188 + j *
100,Height-519);
      }
      CurrentY-=40
      fill(6, 21, 121);
      stroke(15, 21, 71);
      rect(35, CurrentY, 330,50);
      stroke(3, 11, 101);
      strokeWeight(10);
      fill(22, 27, 46);
      rect(65, CurrentY, 270,50);
      strokeWeight(1);
      fill(255,255,255);
      noStroke();
      textSize(26);
      text('POLICE', 90, CurrentY+35);
      text('BOX',260, CurrentY+35);
      textSize(12);
```

```
text('PUBLIC', 200, CurrentY+25);
text('CALL', 209, CurrentY+39);
CurrentY-=30
fill(6, 21, 121);
stroke(15, 21, 71);
rect(65, CurrentY, 270,30);
CurrentY-=20
rect(85, CurrentY, 230,20);  }
```

Rezultatul realizării programului

Concluzii

În urma realizării lucrării de laborator am dobândit cunoștințe practice esențiale privind sinteza scenelor grafice 2D statice, utilizând primitivele grafice simple puse la dispoziție de biblioteca p5.js. Prin utilizarea eficientă a primitivelor grafice (puncte, linii, dreptunghiuri, cercuri etc.) și a funcțiilor de stilizare a fost creată scena grafică, am observat importanța fiecărei primitive în construirea formelor și obiectelor de bază într-o scenă grafică.

Am aprofundat cunoștințele utilizând tehnicile de modificare a atributelor grafice, cum ar fi culoarea, conturul, transparența și umplerea, elemente esențiale în definirea esteticii unei compoziții. De asemenea, am învățat să afișez și să stilizez textul în mod grafic, integrându-l armonios în cadrul scenei, ceea ce reprezintă un pas important în construirea imaginilor.

Experiența obținută în urma realizării lucrării de laborator a contribuit la consolidarea cunoștințelor teoretice și a abilităților practice necesare pentru lucrul cu grafica pe calculator.