UNIVERSITATEA TEHNICĂ A MOLDOVEI

Rotaru Lilia, Oșovschi Mariana,
Cărbune Viorel, Ursu Adriana

GRAFICA PE CALCULATOR

ÎNDRUMAR METODIC PENTRU LUCRĂRILE DE LABORATOR

Chișinău
2025

IV. UTILIZAREA OBIECTELOR 3D ÎN SCENE STATICE

4.1. Fișiere *.OBJ

Formatul fișierului OBJ este unul dintre cele mai importante formate de fișiere în aplicațiile de imprimare 3D și grafică 3D. Este formatul preferat pentru imprimarea 3D multicolor, fiind utilizat pe scară largă ca format de schimb neutru pentru modelele 3D non-animate în aplicațiile grafice.

Fișierul OBJ este un format standard al imaginilor 3D care poate fi exportat și deschis de diverse editoare 3D. Acesta conține obiecte tridimensionale, care includ coordonatele 3D, hărțile de texturi, fețele poligonale și alte informații despre obiect.

Formatul fișierelor OBJ stochează informații despre modelele 3D, poate codifica geometria suprafeței unui model 3D și poate stoca informații despre culoare și textură. Acest format nu stochează nicio informație despre scenă cum ar fi poziția luminii sau animației.

Fișierul OBJ este generat de un software CAD (Computer Aided Design) ca produs finit al procesului de modelare 3D. Extensia fișierului corespunzătoare formatului fișierului OBJ este – „*.OBJ”.

Formatul fișierelor OBJ este open source și neutru, fiind adesea utilizat la partajarea modelelor 3D în aplicațiile grafice, deoarece asigură un suport bun de import și export din aproape toate software-urile CAD. Acesta este de asemenea utilizat ca formatul fișierului pentru imprimarea 3D multicolor, deoarece formatul standard de imprimare 3D STL nu include informații despre culoare și textură.

Formatul fișierelor OBJ a fost creat inițial de Wavefront Technologies privind aplicația sa Advanced Visualizer pentru a stoca obiecte geometrice compuse din linii, poligoane, curbe și suprafețe de formă liberă. Cea mai recentă versiune documentată este v3.0, înlocuind versiunea anterioară v2.11.

Cel mai dominant format din lumea tipăririi 3D este STL. Cu toate acestea, STL este un format de fișiere vechi care, deși este foarte popular, nu ține cont de cerințele moderne. Exactitarea imprimării 3D se apropie de rezoluția de nivel de micron și modelele multicolore

devin din ce în ce mai populare. Formatul fișierelor STL nu poate accepta rezoluțiile mari, deoarece o rezoluție mai mare duce la creșterea dimensiunii fișierului. De asemenea, formatul STL nu este potrivit pentru imprimarea 3D multicolor, deoarece nu acceptă informații despre culoare și textură. În schimb, formatul OBJ poate aproxima geometria suprafeței destul de precis, fără a avea un impact semnificativ asupra dimensiuneii fișierului. Acest lucru este posibil folosind curbele Bezier și o metodă numită NURBS. În plus, formatul OBJ are suport nativ pentru mai multe culori și texturi din același model.

Astfel, utilizarea formatului OBJ are o serie de avantaje în comparație cu utilizarea formatului STL în cazul în care este nevoie de modele cu rezoluție înaltă, multicolor. Pe de altă parte, formatul fișierului OBJ nu este la fel de universal ca și formatul STL. Aproape toate imprimantele 3D acceptă formatul STL. Nu se poate spune același lucru și despre formatul OBJ, chiar dacă se bucură de suport. Prin urmare, dacă este nevoie de a tipări un model 3D monocrom la o imprimantă standard, este preferabil formatul STL.

4.2. Descrierea obiectelor și metodelor principale în lucrul cu fișierele *.OBJ în editorul online p5.js

Biblioteca p5.js este o bibliotecă JavaScript care facilitează procesul de dezvoltare a programelor ce utilizează elemente grafice 2D și 3D și este destinată atât dezvoltatorilor experimentați, cât și celor începători. Biblioteca p5.js este gratuită și open-source.

Biblioteca p5.js are un set complet de funcționalități grafice. Cu ajutorul acestei biblioteci întreaga pagină a browserului este privită ca spațiu de lucru în care pot fi utilizate inclusiv obiecte HTML5 pentru text, video, cameră web și sunet.

Funcțiile de bază:

Încărcarea unui model 3D dintr-un fișier OBJ sau STL

Funcția loadModel(): trebuie plasată în interiorul funcției **preload()**, aceasta permite modelului să se încarce complet înainte de a rula programul până la sfârșit.

Una dintre limitările formatului OBJ și STL este faptul că nu ține cont de scară. Aceasta înseamnă că modelele exportate din diferite programe pot avea dimensiuni diferite. Dacă modelul nu se afișează, se poate apela **loadModel()** cu parametrul de normalizare setat ca și true. Aceasta va redimensiona modelul la o scară adecvată pentru p5. De asemenea, se pot face modificări suplimentare ale dimensiunii modelului cu funcția **scale()**.

La fel, nu este realizat suportul pentru fișierele colorate STL. Fișierele STL colorate vor fi redate fără proprietăți de culoare.

Sintaxa:

```
loadModel(path, normalize, [successCallback], [failureCallback], [fileType]);
loadModel(path, [successCallback], [failureCallback], [fileType]);
```

unde:

path String – calea modelului încărcat;

normalize boolean – dacă este adevărat, se scalează modelul la o dimensiune standard la încărcare;

funcția successCallback Function(p5.Geometry) – funcție care trebuie apelată după încărcarea modelului. Va fi returnat obiectul de tip 3D model. (Opțional);

funcția failureCallback Function(Event) – apelat cu eveniment de eroare dacă modelul nu reușește să se încarce (opțional);

fileType String – extensia fișierului modelului (.stl, .obj) (opțional). Întoarce p5.Geometry – obiect de tip p5.Geometry.

Funcția model() – redă un model 3D pe ecran;

Sintaxa:

```
model(model);
```

unde:

model – p5.Geometrie.

Model 3D încărcat care trebuie redat:

Sketch Files->Upload file

4.3. Crearea scenei statice 3D

Sarcina lucrării de laborator constă în conceperea și construirea scenei statice 3D cu ajutorul editorului online p5.js, utilizând modele de obiecte 3D stocate în fișiere .OBJ create individual sau descărcate din internet.

Pentru exemplificarea desfășurării lucrării de laborator se cere a crea o scenă statică care ar reprezenta o moară de vânt îngrădită de un gard în ograda căreia se vor afla câteva animale, un arbore și un stog de fân. Modelele 3D ale acestor obiecte pot fi create sau importate din repozitoriul 3D Warehouse care poate fi accesat din meniul de bază al editorului grafic 3D Google SketchUp ***Window->3D Warehouse***, apoi editate după necesitate, după cum este arătat în figura 4.1.

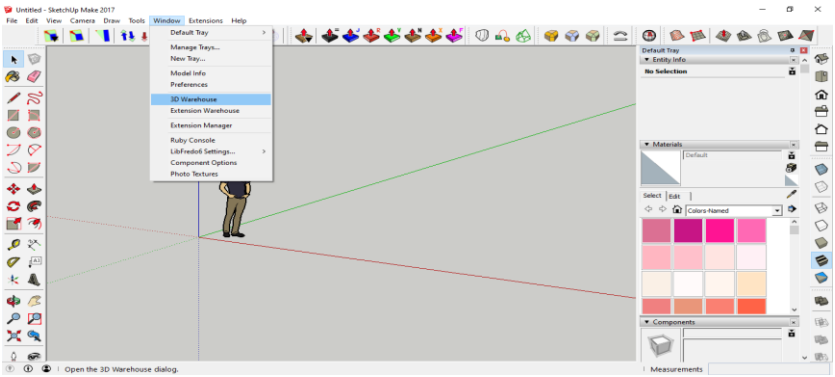

Figura 4.1. Fereastra de lucru a repozitoriului 3D Warehouse

Pentru a importa modelul selectat în sketch-ul curent, utilizatorul trebuie să dispună de un cont Google Account și să fie autentificat în mediul 3D Warehouse. După ce a fost downloadat modelul/modelele necesare, acestea pot fi editate după necesitate și exportate ca fișiere .OBJ prin intermediul plug-inului ***LIPID OBJ Exporter***, care poate fi instalat din meniul *Window->Extension Warehouse*. În figura 4.2, în câmpul de căutare al căruia indicăm cuvântul-cheie OBJ, ca răspuns sunt afișate plugin-urile ce satisfac

criteriile de căutare printre care poate fi observat și plugin-ul căutat *LIPID OBJ Exporter*, după cum este arătat în figura 4.3.

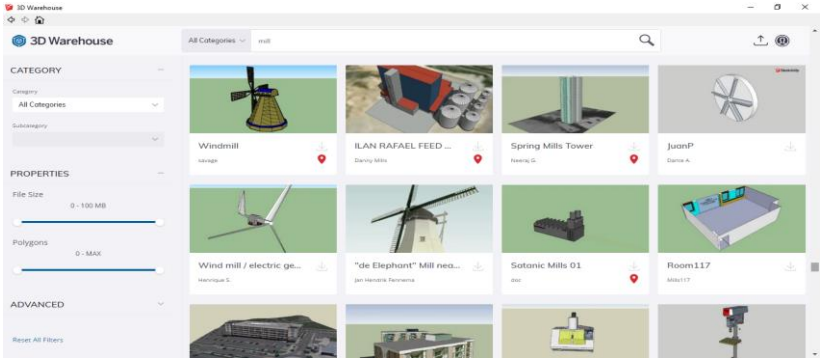

Figura 4.2. Căutarea modelelor 3D în repozitoriul 3D Warehouse

Pentru a instala plugin-urile, utilizatorul trebuie să dispună de un cont Google Account și să fie autentificat în repozitoriul online *Extension Warehouse*.

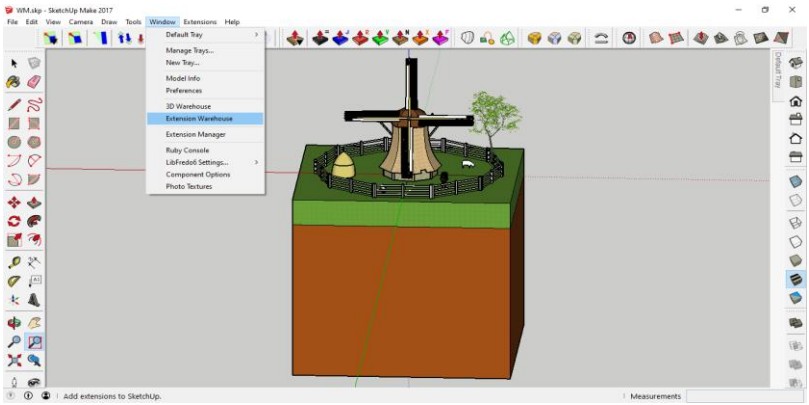

Figura 4.3. Accesarea opțiunii Extension Warehouse din bara de meniu

În figurile de mai jos sunt arătați pașii care trebuie urmați pentru crearea modelului *.obj.

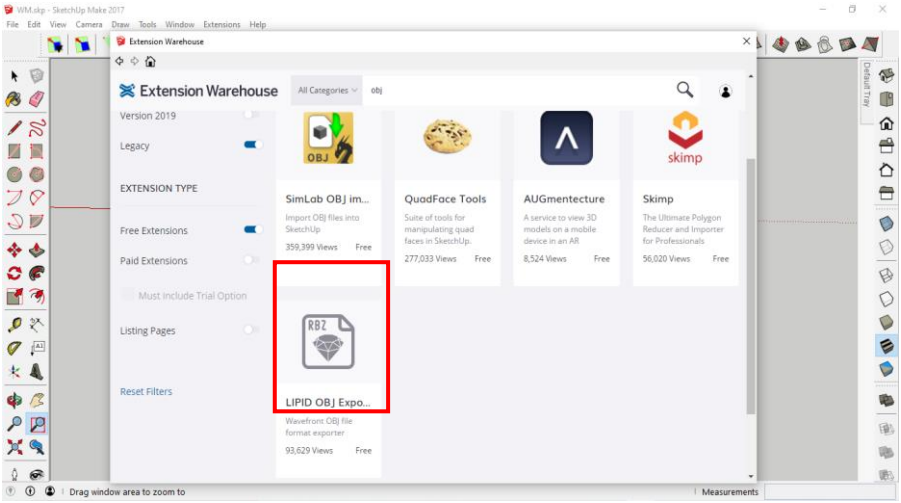

Figura 4.4. Fereastra de căutare și instalare a plugin-urilor din repozitoriul online Extension Warehouse

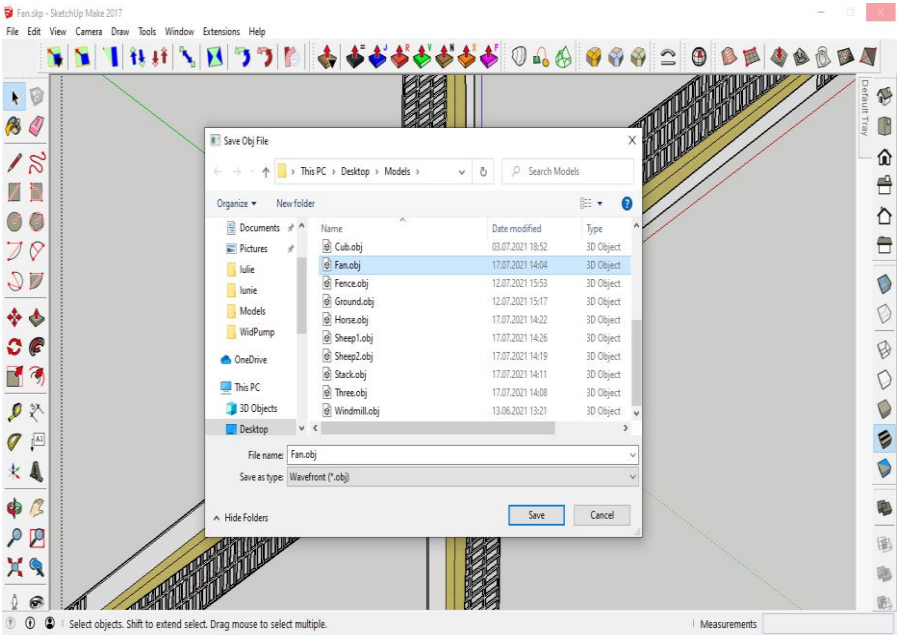

Figura 4.5. Fereastra de salvare în format .obj a modelelor

3D ale obiectelor din scenă

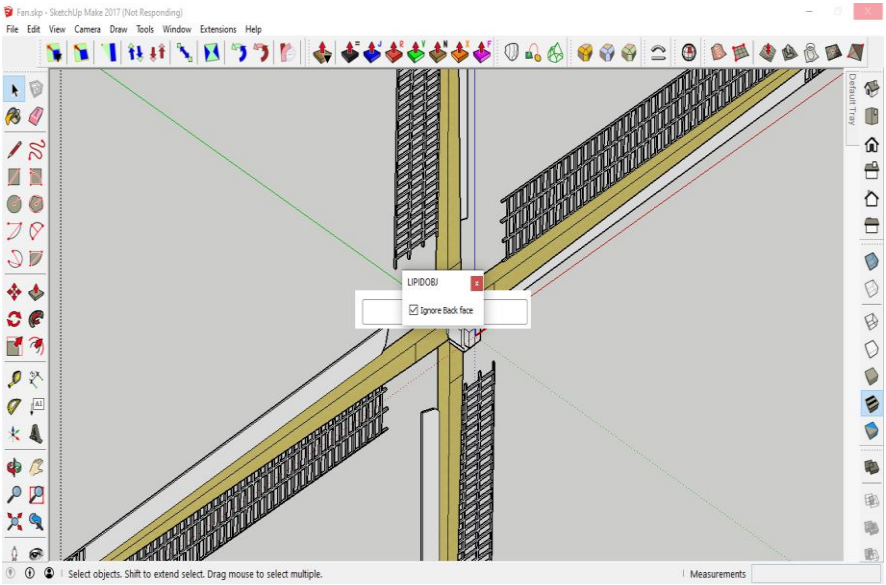

Figura 4.6. Opțiunea de ignorare a fețelor plugin-ului LIPIDOBJ la exportarea modelelor 3D în format .obj

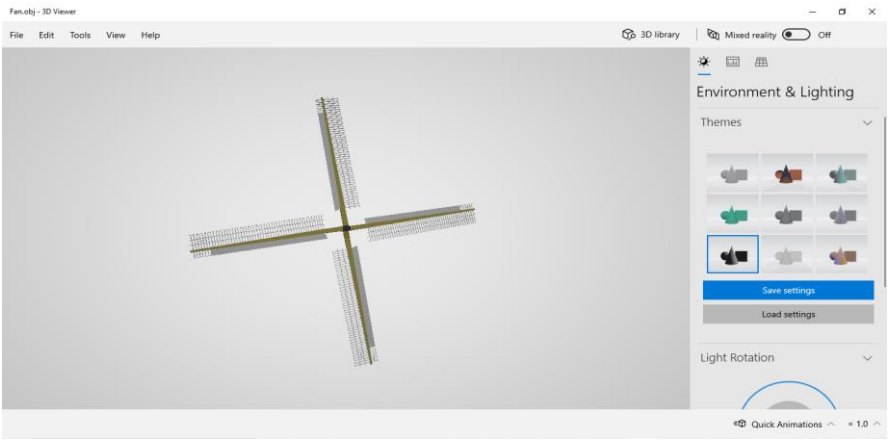

Figura 4.7. Vizualizarea conținutului fișierelor .OBJ cu ajutorul aplicației standard din cadrul sistemului de operare Windows 10–3D Viewer

Lucrarea de laborator nr. 4
Tema: CREAREA SCENEI STATICE 3D

Obiectivele lucrării:

1. Crearea unui model 3D static – se utilizează un editor grafic compatibil cu exportul în format .obj (de exemplu, Blender, SketchUp sau Warehouse).

2. Exportul correct al modelului – în formatul .obj și importarea acestuia într-un mediu de programare p5.js.

3. Adăugarea de primitive grafice 3D – în p5.js pentru completarea și personalizarea scenei 3D.

4. Explorarea funcționalităților p5.js pentru lucrul cu obiecte 3D prin adăugarea atributelor de culoare și lumină.

5. Aplicarea conceptelor de grafică pe calculator – poziționarea și structurarea scenei tridimensionale.

6. Crearea unei scene finale care să combine modelul 3D importat și elementele grafice adăugate direct în p5.js pentru un efect vizual coerent.

Numărul de ore necesare pentru realizare – 4 ore academice.

Scopul lucrării: obţinerea cunoştinţelor practice privind sinteza scenelor grafice 3D statice, utilizând funcțiile standard de reprezentare a modelelor 3D din biblioteca p5.js. Dezvoltarea competențelor de modelare și vizualizare a obiectelor 3D prin utilizarea editorilor grafici și integrărea modelelor în formatul *.obj în medii interactive bazate pe JavaScript.

Sarcina lucrării:

1. Elaborați un obiect cu extensia ***.obj** în care să creați o scenă 3D statică conform variantei indicate în tabelul 4.1. Pentru crearea scenei pot fi utilizate obiecte grafice 3D existente în repozitoriul 3D Warehouse.

2. Elaborați un program în care sintetizați o scenă 3D statică care ar conține cel puțin 5 obiecte, utilizând funcțiile standard de

reprezentare a modelelor 3D din biblioteca p5.js, în scena creată importați obiectul *.obj creat în punctul 1 al sarcinii.

Tabelul 4.1. Variante pentru realizarea lucrării de laborator

Nr.	***Obiect 3D***	***Model***
1	Carusel	
2	Far maritim	
3	Automobil blindat	
4	Moară de apă	

Nr.	*Obiect 3D*	*Model*
5	Avion cu elice	
6	Morișcă de vânt	
7	Turbină eoliană	
8	Elicopter	
9	Submarin	

Nr.	*Obiect 3D*	*Model*
10	Catapultă	

Exemplu:

```
let fr = 30;
let obj;
function preload() {
  mill = loadModel('Windmill.obj');
  fan = loadModel('Fan.obj');
  gnd = loadModel('Ground.obj');
  sheep1 = loadModel('Sheep1.obj');
  sheep2 = loadModel('Sheep2.obj');
  horse = loadModel('Horse.obj');
  stack = loadModel('Stack.obj');
  fence = loadModel('Fence.obj');
  three = loadModel('Three.obj'); }
function setup() {
  createCanvas(400, 400, WEBGL);
  normalMaterial();
  frameRate(fr);
  angleMode(DEGREES);}
function draw() {
  orbitControl();
  background(50);
  pointLight(255, 255, 255, 100000, 100000,-
100000);
  noStroke();
```

```
push();
scale(0.005);
translate(0, 0, 0);
model(mill);
pop();
push();
scale(0.005);
translate(0, 3000, 13000);
rotateX(15);
model(fan);
pop();
push();
scale(0.005);
model(gnd);
pop();
push();
scale(0.005);
translate(-3000, 15000, 0);
rotateZ(-10);
model(sheep1);
pop();
push();
scale(0.005);
translate(10000, 0, 0);
rotateZ(180);
model(sheep2);
pop();
push();
scale(0.005);
rotateZ(-15);
translate(7000, 12000, 0);
model(horse);
pop();
push();
```

```
scale(0.005);
translate(-12000, -2000, 0);
model(stack);
pop();
push();
scale(0.005);
translate(0, 0, 0);
model(fence);
pop();
push();
scale(0.005);
translate(15000, -15000, 0);
model(three);
pop(); }
```

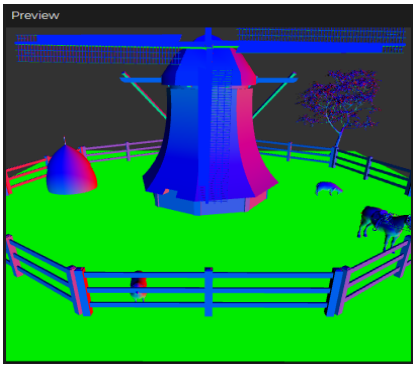

Figura 4.8. Rezultatul execuției programului

Criterii de evaluare

1. Corectitudinea modelării 3D în editorul grafic (20%) – dacă obiectul 3D realizat este conform cerințelor și bine definit, dacă exportarea în formatul .obj s-a realizat corect și fără pierderi de date.

2. Integrarea modelului 3D în p5.js (25%) – dacă modelul importat este funcțional în p5.js fără erori de afișare, dacă poziționarea modelului în scena 3D este corectă și adecvată.

3. Adăugarea și utilizarea primitivelor grafice în p5.js (25%) – dacă în scenă au fost adăugate primitive grafice 3D care contribuie la compoziția generală, dacă primitivelor li s-au atribuit poziționări și dimensiuni adecvate pentru a crea o scenă coerentă.

4. Respectarea cerințelor și claritatea codului (10%) – dacă codul este clar, bine organizat și documentat, respectând cerințele tehnice specificate. Comentariile explicative sunt incluse unde este necesar pentru a clarifica funcționalitatea. Dacă scena este estetică și bine structurată, cu o compoziție echilibrată, sau au fost aplicate concepte de bază de grafică (iluminare, umbre, perspectivă) pentru o experiență vizuală plăcută.

5. Evaluarea cunoștințelor (10%) – explicații privind procesul de realizare a lucrării, ceea ce poate include descrierea funcțiilor principale și a logicii utilizate.

6. Respectarea cerințelor și termenului de prezentare finală a proiectului – (10%).

Întrebări de autoevaluare a cunoștințelor

1. Ce este un model 3D static și prin ce diferă de unul animat?
2. Explicați pașii necesari pentru a exporta un model 3D în format .obj și a-l importa într-un proiect p5.js.
3. Ce primitive grafice 3D în p5.js cunoașteți? Enumerați câteva exemple și aplicații posibile.
4. Cum se realizează rotirea, translația și scalarea unui obiect 3D în p5.js?
5. Care sunt avantajele utilizării p5.js pentru redarea scenelor 3D statice?
6. Cum puteți ajusta iluminarea în scena 3D p5.js pentru a obține un efect realistic?
7. Ce funcții și metode ați folosit pentru a adăuga elemente grafice 3D în p5.js și cum au contribuit ele la aspectul scenei finale?

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.