Transformări grafice 2D

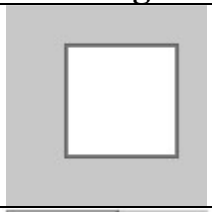
Transformări grafice 2D în biblioteca grafică p5js:

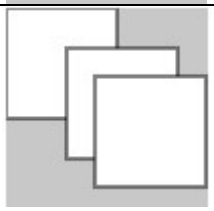
- (translate)
- (scale)
- (rotate)
- (shear)

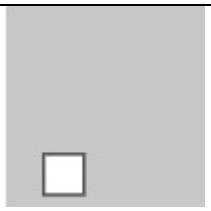
Translarea

Funcția **translate()** permite deplasarea obiectelor în orice locație din fereastră. În tabelul 2.1 este prezentată sintaxa translației.

Tabelul 2.1. Sintaxa translației

Cod	Rezultat grafic
`translate(30, 20);` `rect(0, 0, 55, 55);`	

`rect(0, 0, 55, 55); // Draw rect at original 0,0`
`translate(30, 20);`
`rect(0, 0, 55, 55); // Draw rect at new 0,0`
`translate(14, 14);`
`rect(0, 0, 55, 55); // Draw rect at new 0,0`
	

`function draw() {`
`  background(200);`
`  rectMode(CENTER);`
`  translate(width / 2, height / 2);`
`  translate(p5.Vector.fromAngle(millis() / 1000, 40));`
`  rect(0, 0, 20, 20);`
`}`
	

În exemplul de mai jos este prezentată executarea translației prin cod:

```
let x = 0;
let y = 0;
let dim = 80.0;

function setup() {
  createCanvas(720, 400);
  noStroke();
}
```

```
function draw() {
  background(102);

  // Animează creșterea valorii x

  x = x + 0.8;

  // Dacă forma iese din grilă, resetați poziția

  if (x > width + dim) {
    x = -dim;
  }

// Chiar dacă comanda noastră „rect” desenează forma cu
centrul său la
// origine, translația o mută în noua poziție x și y

  translate(x, height / 2 - dim / 2);
  fill(255);
  rect(-dim / 2, -dim / 2, dim, dim);

// Transformările se acumulează. Observați cum acest „rect”
// se mișcă de două ori mai repede decât celălalt, dar are
// același parametru pentru valoarea axei x

  translate(x, dim);
  fill(0);
  rect(-dim / 2, -dim / 2, dim, dim);
}
```

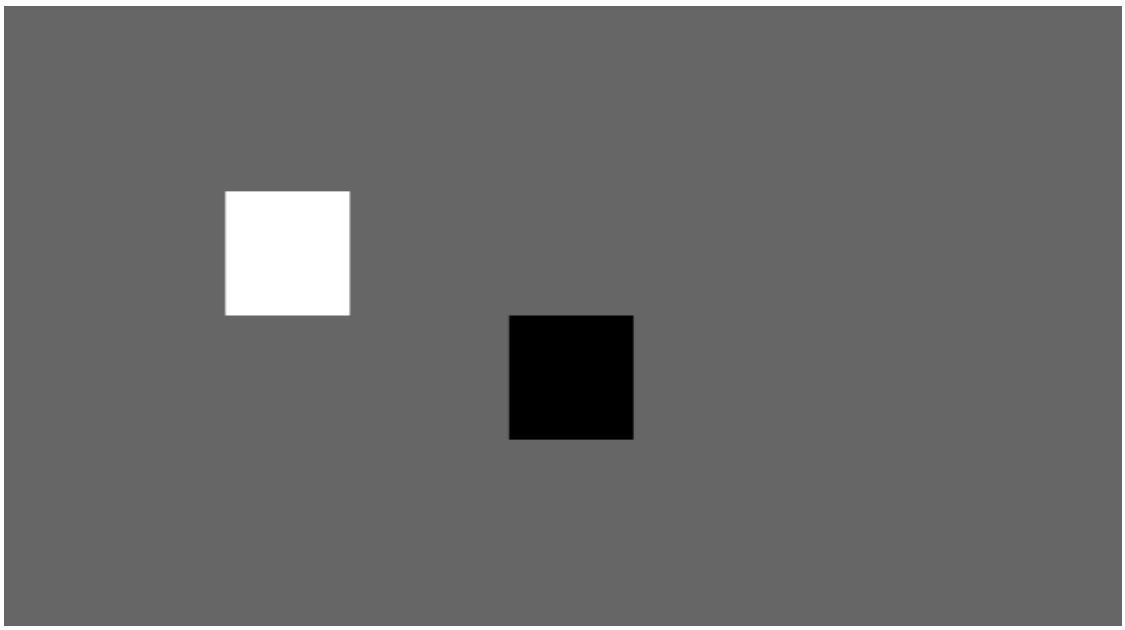


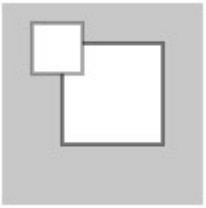
Figura 2.1. Rezultatul executării codului (translării)

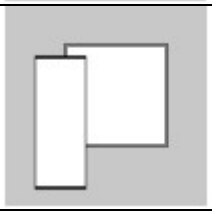
Scalarea

Funcția **scale()** transformare care redimensionează un element în planul 2D. Deoarece valoarea scalării este definită de un vector, aceasta poate modifica dimensiunile orizontale și verticale la scări diferite. În tabelul 2.2 este prezentată sintaxa scalării.

Tabelul 2.2. Sintaxa scalări

Cod	Rezultat grafic

`rect(30, 20, 50, 50);`
`scale(0.5);`
`rect(30, 20, 50, 50);`
	

`rect(30, 20, 50, 50);`
`scale(0.5, 1.3);`
`rect(30, 20, 50, 50);`
	

Parametrii pentru funcția **scale()** sunt valori specificate ca procente zecimale. De exemplu, metoda scala de apel (2.0) va crește dimensiunea formei cu 200 la sută. Obiectele se întind începând de la origine. Exemplu de mai jos arată cum se acumulează transformările și, de asemenea, cum interacționează scalarea și translarea în funcție de ordinea lor.

```
let a = 0.0;
let s = 0.0;

function setup() {
  createCanvas(720, 400);
  noStroke();

// Desenați toate dreptunghiurile din centrul lor

  rectMode(CENTER);
}

function draw() {
  background(102);

 // Creșteți „a” și apoi animați „s” cu
 // o mișcare ciclică prin găsirea cosinusului „a”

  a = a + 0.04;
  s = cos(a) * 2;
```

```
 // Translați dreptunghiul de la origine la mijlocul
 // grlei, apoi scalați-l cu „s”

  translate(width / 2, height / 2);
  scale(s);
  fill(51);
  rect(0, 0, 50, 50);

 // Translarea și scala se acumulează, această translare
 // deplasează al doilea dreptunghi mai la dreapta decât
primul
 // și scara se dublează. Rețineți că cosinusul este
 // făcând „s” atât negativ, cât și pozitiv, astfel se
ciclează
 // de la stanga la dreapta.

  translate(75, 0);
  fill(255);
  scale(s);
  rect(0, 0, 50, 50);
}
```

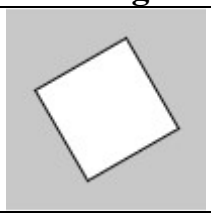
Figura 2.2. Rezultatul executării codului (combinații dintre scalare și translare)

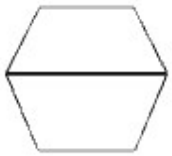
Rotirea

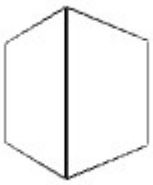
Funcția de rotire **rotate()** definește o transformare care rotește un element în jurul unui punct fix din planul 2D, fără a-l deforma. Rezultatul este rotirea în jurul unei axe.

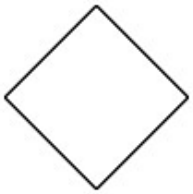
Tabelul 2.3. Sintaxa rotirii

Cod	Rezultat grafic

`translate(width / 2, height / 2);`
`rotate(PI / 3.0);`
`rect(-26, -26, 52, 52);`
	

`function setup() {`
`  createCanvas(100, 100, WEBGL);`
`}`
`function draw() {`
`  background(255);`
`  rotateX(millis() / 1000);`
`  box();`
`}`
	

`function setup() {`
`  createCanvas(100, 100, WEBGL);`
`}`
`function draw() {`
`  background(255);`
`  rotateY(millis() / 1000);`
`  box();`
`}`
	

`function setup() {`
`  createCanvas(100, 100, WEBGL);`
`}`
`function draw() {`
`  background(255);`
`  rotateZ(millis() / 1000);`
`  box();`
`}`
	

Exemplu. Rotirea unui pătrat în jurul axei Z. Pentru a obține rezultatul pe care îl dorim, parametrii unghiului funcției de rotație trebuie să fie valori cuprinse între 0 și PI * 2 (TWO_PI care este de aproximativ 6,28). Putem utiliza și valori unghiulare în grade (0 – 360), folosind metoda radians () pentru a converti valorile. De exemplu: rotate (radians (90)) este identic cu declarația rotate (PI / 2). În exemplu de mai jos, în fiecare secundă pară se execută o mișcare de rotație. În secundele impare, rotația se produce CW (în sensul acelor de ceasornic) și CCW (în sens invers acelor de ceasornic) la viteza determinată de ultima valoare a jitterului (tremurului).

```
let angle = 0.0;
let jitter = 0.0;

function setup() {
  createCanvas(720, 400);
  noStroke();
  fill(255);

  // Desenați dreptunghiul din centru care va fi și
  // rotația în jurul acelui centru

  rectMode(CENTER);
}

function draw() {
  background(51);

  // în secunde par (0, 2, 4, 6...) adăugați jitter(tremur) la rotație

  if (second() % 2 === 0) {
    jitter = random(-0.1, 0.1);
  }

  // măriți valoarea unghiului folosind
  // cea mai recentă valoare de jitter(tremur)

  angle = angle + jitter;

 // utilizați cosinusul pentru a obține
 // o mișcare lină CW și CCW atunci când nu vibrați

  let c = cos(angle);

  // mutați forma în centrul grilei

  translate(width / 2, height / 2);

  // aplicați rotația finală

  rotate(c);
  rect(0, 0, 180, 180);
}
```

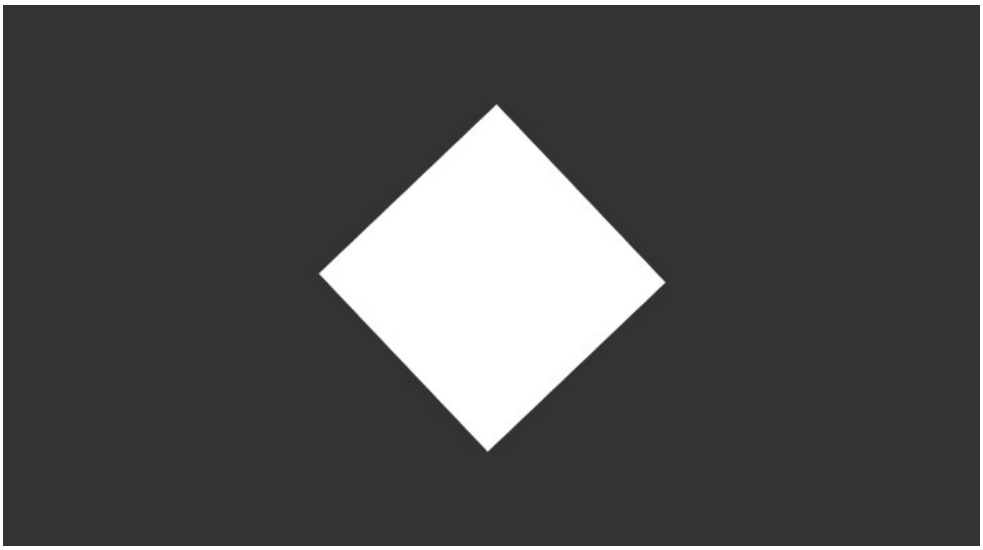


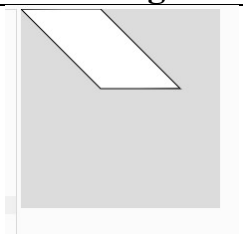
Figura 2.3. Rezultatul executării codului (rotire cu adăugare de tremur)

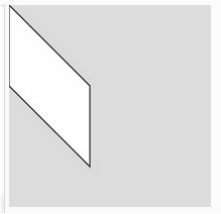
Forfecarea

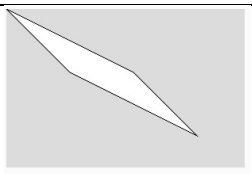
Forfecarea **shear()** este o transformare care produce distorsionarea obiectului transformat.

Tabelul 2.4. Sintaxa forfecării

Cod	Rezultat grafic

`function setup() {`
`  createCanvas(200, 200);`
`}`
`function draw() {`
`  background(220);`
`  shearY(PI / 4.0);`
`  rect(0, 0, 80, 80);`
`}`
	

`function setup() {`
`  createCanvas(200, 200);`
`}`
`function draw() {`
`  background(220);`
`  shearY(PI / 4.0);`
`  rect(0, 0, 80, 80);`
`}`
	

`function setup() {`
`  createCanvas(300, 200);`
`}`
`function draw() {`
`  background(220);`
`  shearX(PI / 4.0);`
`  shearY(PI / 4.0);`
`  rect(0, 0, 80, 80);`
`}`
	

Forfecarea o formă de transformare în jurul axei *x* sau *y* cu valoarea specificată a parametrul unghiului. Unghiurile ar trebui specificate în angleMode curent. Obiectele sunt întotdeauna forfecate în jurul poziției lor relative față de origine, iar numerele pozitive forfecă obiectele în sensul acelor de ceasornic.

Transformările se aplică la tot ceea ce se întâmplă după, iar apelurile ulterioare la funcție acumulează efectul. De exemplu, apelarea shearX(PI/2) și apoi shearX(PI/2) este aceeași cu shearX(PI). Dacă shearX() este apelată în cadrul draw(), transformarea este resetată când bucla începe din nou.

Din punct de vedere tehnic, shearX() înmulțește matricea de transformare curentă cu o matrice de rotație. Această funcție poate fi controlată în continuare de funcțiile push() și pop().

Lucrare de laborator nr. 2
Tema: Transformări grafice 2D

Scopul: Implementarea transformărilor grafice asupra unei scene 2D utilizând setul de funcționalități a bibliotecii (JavaScript) p5.js.

Sarcina: Elaborați un program pentru efectuarea transformărilor grafice 2D utilizând **rotate()**, **scale()**, **translate(), shear()**.

Transformările v-or fi aplicate asupra scenei 2D create la lucrarea de laborator nr. 1.

Exemple de program pentru fiecare transformare puteți găsi mai sus în tabelele 2.1, 2.2, 2.3 și 2.4.

Tabelul 2.4. Variantele pentru realizarea lucrării de laborator

Varianta	Unghiul de rotire	Coeficientul de scalare	Deplasarea X și Y	Forfecarea
1	320	0.5	10,10	QUARTER_PI
2	70	1.2	20,10	HALF_PI
3	30	2.1	15,10	HALF_PI+ QUARTER_PI
4	25	0.8	100,100	PI+ QUARTER_PI
5	100	1.3	90,85	PI+ HALF_PI
6	110	1.4	15,15	PI+ HALF_PI +QUARTER_PI
7	20	0.6	80,80	PI / 3.0
8	49	1.1	45,40	PI / 4.0
9	95	0.8	20,25	PI / 5.0
10	64	1.9	70,70	PI / 6.0
11	38	0.2	35,35	PI / 7.0
12	128	2.3	40,45	PI / 8.0
13	300	1.47	100,110	0.31
14	256	0.3	90,95	0.47
15	49	1.3	20,10	0.79
16	60	1.29	10,20	1.26
17	83	1.7	40,20	1.73
18	39	0.4	85,100	2.20
19	26	1.23	30,35	2.51
20	74	1.97	20,30	2.83
21	91	2.3	105,125	3.30
22	90	1.32	110,130	3.61
23	95	1.90	100,90	4.08
24	44	1.5	120,115	4.56
25	79	0.9	20,40	5.03