UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICA, MICROELECTRONICA

GRAFICA PE CALCULATOR

ÎNDRUMĂR METODIC PENTRU LUCRĂRI DE LABORATOR
(draft)

2022

Chişinău 2022

Introducerea în biblioteca grafică p5.js

Ce este p? 5.js

P5.js este o bibliotecă scrisă în limbajul de programare JavaScript, utilizată pentru crearea și vizualizarea imaginilor interactive cu ajutorul primitivelor grafice simple. P5.js permite crearea graficii pe calculator folosind un limbaj de programare. Aceasta permite integrarea simplă a codurilor scrise în pagini web prin adăugarea codului scris întru-un document HTML.

P5.js este gratis, open-source și independent de platformă, deci aplicațiile pot fi rulate pe orice sistem de operare, de asemenea p5.js are o familie mare de limbaje și medii programare înrudite, aceste sunt prezentate în figura 1.

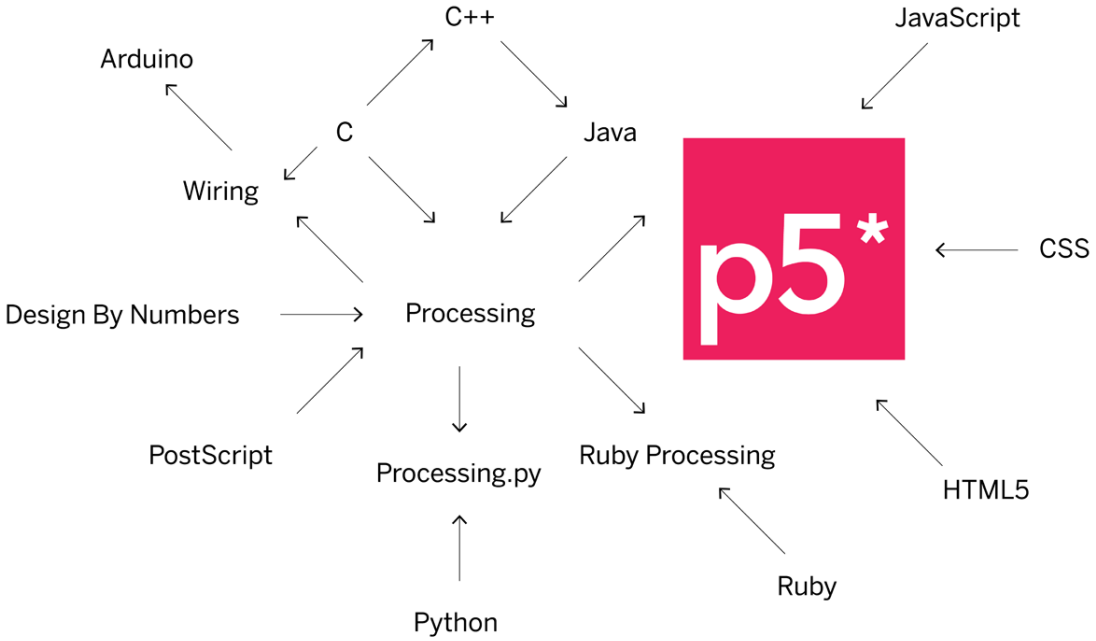


Figura 1.1. Limbaje și medii de programare înrudite cu p5.js

Creare unui program simplu în P5.js

Înainte de a începe crearea programelor proprii trebuie să știm că orice figură creată în mediul p5.js este legată de sistemul de coordonate, originea sistemului de coordonate în orice program este colțul dis stânga sus al ecranului. Axa verticală se numește axa Y, iar cea orizontală axa X. Creșterea valorilor pentru coordonatele x și y sunt prezentate în figura 2.

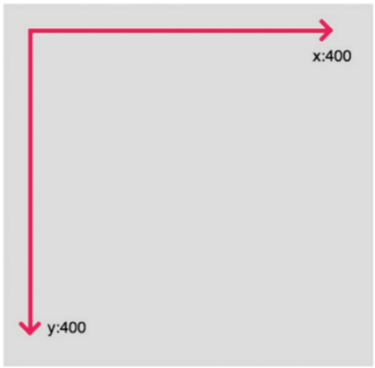


Figura 1.2. Sistemul de coordonate în mediul p5.js

Pentru crearea unui program simplu în p5.js trebuie să utilizăm următorul șablon:

```
function setup(){
    createCanvas(400,400);
}
function draw(){
    background(220);}
```

Rezultatul programului și exemplul editorului online p5.js este arătat în figura 3.

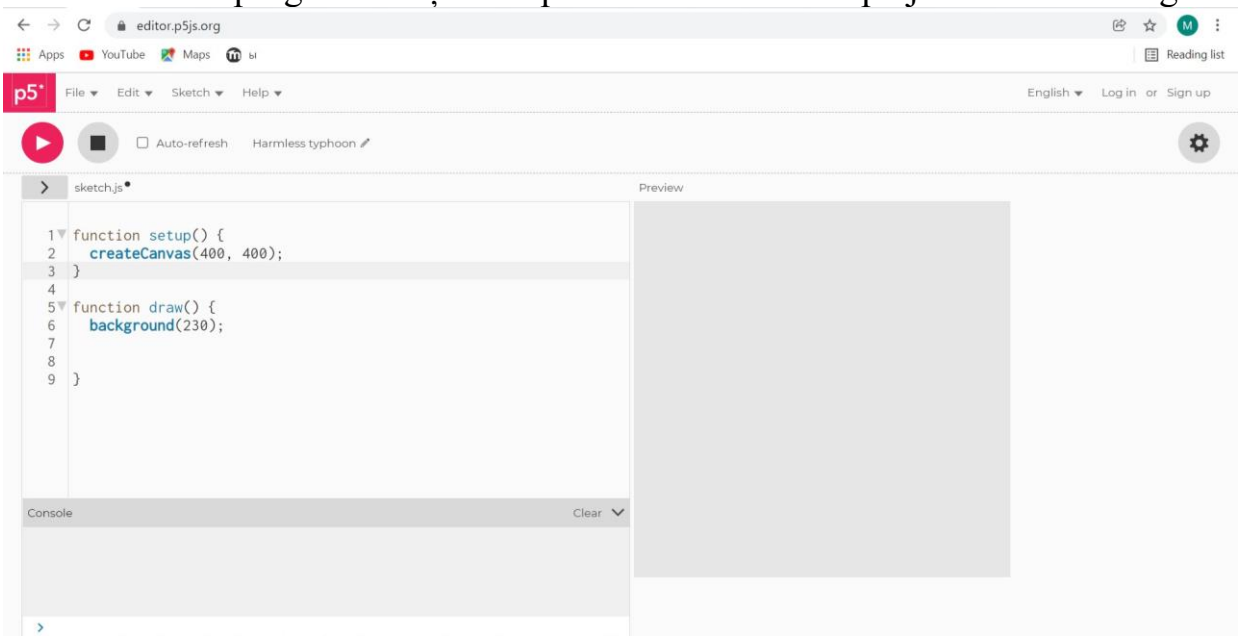


Figura 1.3. Fereastră de lucru în mediul p5.js

Funcția setup() – se apelează o singură dată la începutul programului. Este folosită pentru setarea proprietăților inițiale a mediului de lucru, cum ar fi dimensiunea și culoarea ecranului, precum și pentru încărcarea fișiere multimedia la lansarea programul, cum ar fi imagini și fonturi. Poate exista o singură funcție setup() per program și nu ar trebui apelată după execuția inițială.

Notă: Variabilele declarate în setup() nu sunt disponibile în alte funcții, inclusiv draw ().

Exemplu:

```
function setup() {
createCanvas();
}
```

createCanvas – creează un element canvas (pânză) și setează dimensiunea acestuia în pixeli. Această metodă ar trebui apelată o singură dată la începutul programului. Apelarea createCanvas de mai multe ori într-un cod va avea ca rezultat un comportament foarte imprevizibil. Dacă doriți mai mult de o pânză de desen, puteți utiliza createGraphics.

Sintaxă metodei este următoarea:

```
createCanvas(w, h, [renderer])
```

unde:

w - număr: lățimea pânzei;

h - număr: înălțimea pânzei;

constanta renderului: P2D - dacă originea sistemului de coordonate este în colțul stâng sus a ecranului sau WEBGL – dacă originea sistemului de coordonate este în centrul pânzei este specific pentru grafica 3D (opțională).

Dacă createCanvas () nu este utilizat în program pânzei o să-i fie atribuită valoarea implicită 100x100.

Funcția draw () se apelează imediat după setup(), execută în continuu rândurile de cod care sunt incluse în corpul său, până la săvârșitul programului sau până la apelarea noLoop().

Notă: dacă în setup () este apelată funcția noLoop(), funcția draw() se va executa o singură dată.

Sintaxa funcției este:

```
function draw() {
---------
}
```

Funcția background () setează culoarea utilizată ca fonul canvas-ului. Implicit fonul este transparent. Această funcție de obicei se utilizează în draw() pentru a șterge fereastra de afișare la începutul fiecărui cadru dar, poate fi utilizată și în interiorul funcției setup(), pentru a seta fundalul pe primul cadru al animației sau dacă fundalul trebuie setat o singură dată.

Culoarea este specificată în RGB, HSB sau HSL, în dependență de colorMode. (Implicit modul este - RGB, deci fiecare valoare este în diapazonul 0 ÷ 255).

Sintaxa funcției este:

```
background(color)
background(colorstring, [a])
background(gray, [a])
background(v1, v2, v3, [a])
background(values)
background(image, [a])
```

color:	orice valoare creată cu ajutorul funcției color();
colorstring String:	un string (denumirea culorii în engleză), formatele posibile: un număr întreg rgb () sau rgba (), procent rgb () sau rgba (), 3-cifre hexazecimale, 6-cefre hexazecimale;
a (număr):	opacitatea fundalului în raport cu gama de culori curentă (implicit 0-255) (obțional);
gray (număr):	specifică valoarea între alb și negru;
v1 (număr)	specifică valoarea roșie sau valoarea nuanței (în dependență de gama curentă de culori);
v2 (număr):	specifică valoarea verde sau valoarea saturației (în dependență de gama curentă de culori)
v3 (număr):	specifică valoarea albastră sau valoarea luminozității (în dependență de gama curentă de culori)
values Number []:	un masiv care, conține componentele roșie, verde, albastră și alfa a culorilor;
image:	imaginea creată utilizând loadImage() sau createImage(), pentru a fi setată ca fon;

Capitolul 1
1.1 Crearea primitivelor grafice 2D simple

Primitivele grafice simple reprezintă figurile geometrice care pot fi create cu ajutorul funcțiilor din biblioteca grafică P5.js. Cele mai simple primiteve grafice sunt primitivele grafice 2D, în figura 1.1 arătată corespunderea punctelor figurilor geometrice și parametrilor care trebuie indicate ca argumentele funcțiilor în codul programului.

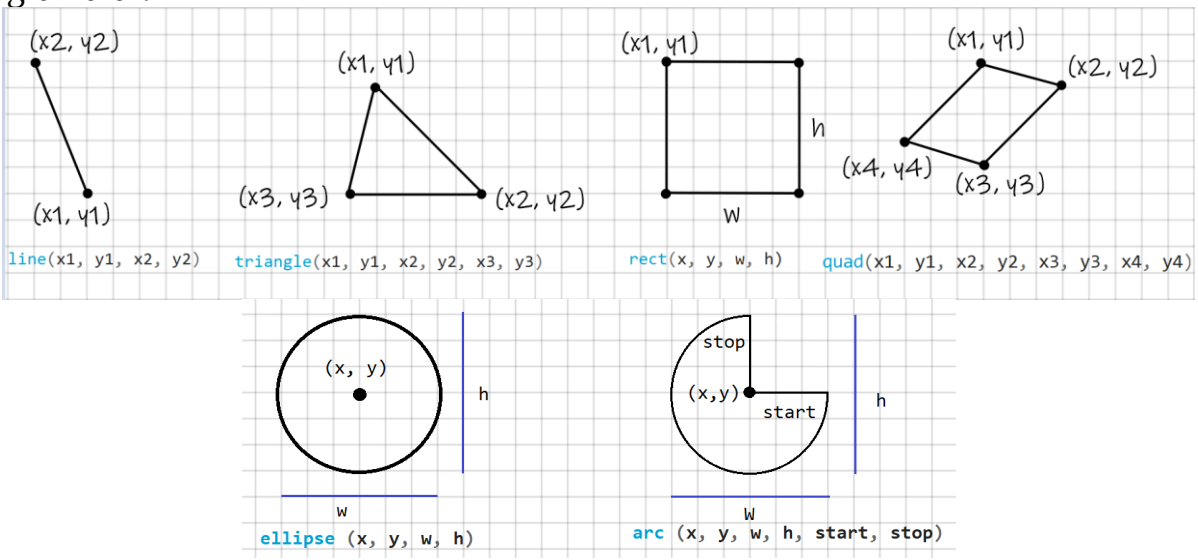


Figura 1.1. Relația dinte punctele geometrice și parametrii funcțiilor P5.js

Funcția arc (): desenează un arc pe ecran. Dacă este apelat doar cu x, y, w, h, start și stop, arcul va fi desenat și umplut ca un segment de cerc deschis. Dacă este specificat parametru *mod*, arcul va fi umplut ca un semicerc deschis (OPEN), un semicerc închis (CHORD) sau un segment circular închis (PIE). Sursa poate fi schimbată funcția ellipseMode(). Diferența dintre aceste regimuri de desenare este arătată in figura 1.2.

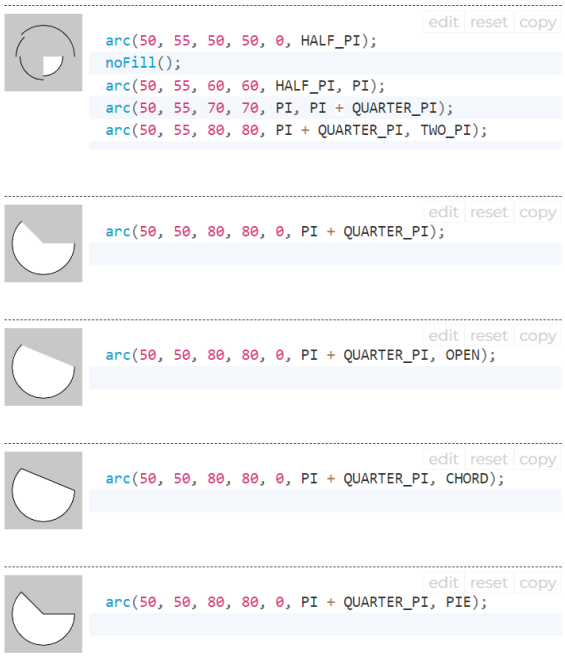


Figura 1.2. Exemple de desenare a arcurilor

Arcul întotdeauna este desenat în sensul acelor ceasornicului punctul de început (start) și sfârșit (stop) pot folosi constantele p5.js conform valorilor indicate în circumferința prezentată în figura 1.3. Dacă punctele start și stop cad în același loc, se va desena o elipsă (cerc) completă. Rețineți că axa Y crește în direcția în jos, astfel încât unghiurile sunt măsurate în sensul acelor de ceasornic din direcția X pozitivă.

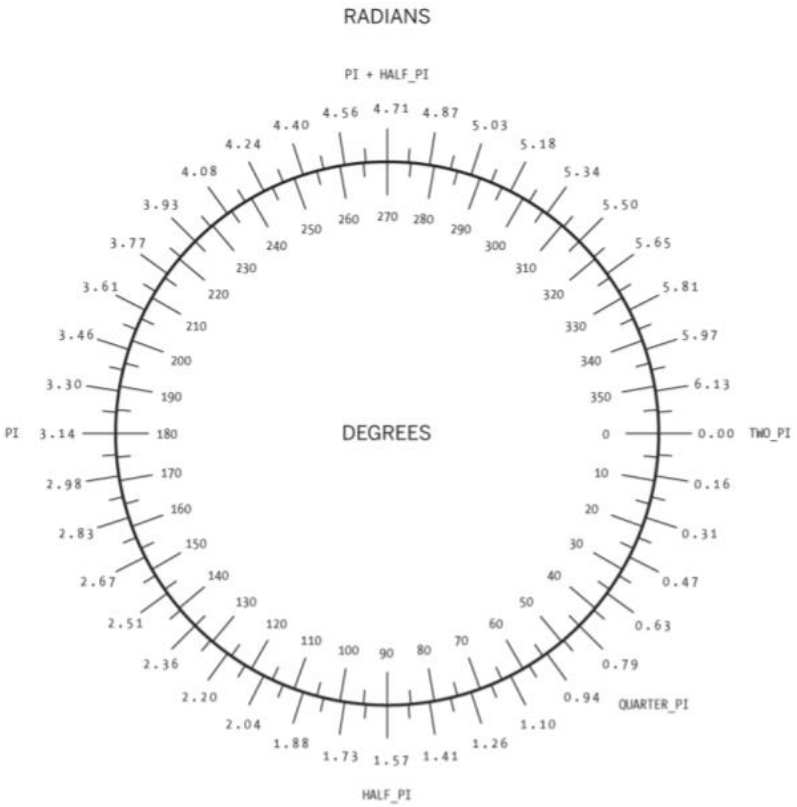


Figura 1.3. Relația dintre constante, radiani și grade în p5.js

Sintaxa:

```
arc (x, y, w, h, start, stop, [mode], [detail])
```

parametrii:
x (număr întreg): coordonata x a arcului;
y (număr întreg): coordonata y a arcului;
w (număr întreg): lățimea arcului;
h (număr întreg): înnălțimea arcului;
start (număr întreg): unghiul de început a arcului;
stop (număr întreg): unghiul de svîrșit a arcului;
mode constanta: parametru care determină modul în care este desenat arcul. COORD, PIE sau OPEN (opțional);
detail (număr întreg): parametru opțional numai pentru modul WebGL;

Funcția ellipse(): desenează o elipsă (ovală) pe ecran. O elipsă cu lățime și înălțime egale este un cerc. În mod implicit, primii doi parametri specifică coordonatele centrului figurei, iar al treilea și al patrulea parametri specifică lățimea și înălțimea. Dacă nu este specificată înălțimea, valoarea lățimii este utilizată atât pentru lățime, cât și pentru înălțime. Dacă se specifică o înălțime sau o lățime negativă, se ia valoarea absolută. Sursa poate fi schimbată folosind funcția ellipseMode().

Sintaxa:

```
ellipse(x, y, w, [h])
ellipse (x, y, w, h, detail)
```

parametrii:
x (număr întreg): coordonata x a centrului figurei;
y (număr întreg): coordonata y a centrului figurei;
w (număr întreg): lățimea elipssei;
h (număr întreg): înnălțimea elipsei;
detail (număr întreg): parametru opțional numai pentru modul WebGL;

Funcția circle(): desenează un cerc pe ecran. Această funcție este un caz particular al funcției elipse() unde lățimea și înălțimea elipsei sunt aceleași. Înălțimea și lățimea elipsei corespund diametrului cercului. În mod implicit, primii doi parametri stabilesc coordonatele centrului cercului, al treilea - diametrul cercului

Sintaxa:

```
circle(x, y, d)
```

parametrii:
x (număr întreg): coordonata x a centrului figurei;
y (număr întreg): coordonata y a centrului figurei;
d (număr întreg): diametrul cercului;
În figura 1.4 este arătat un exemplu de utilizare a funcțiilor ellipse și circle.

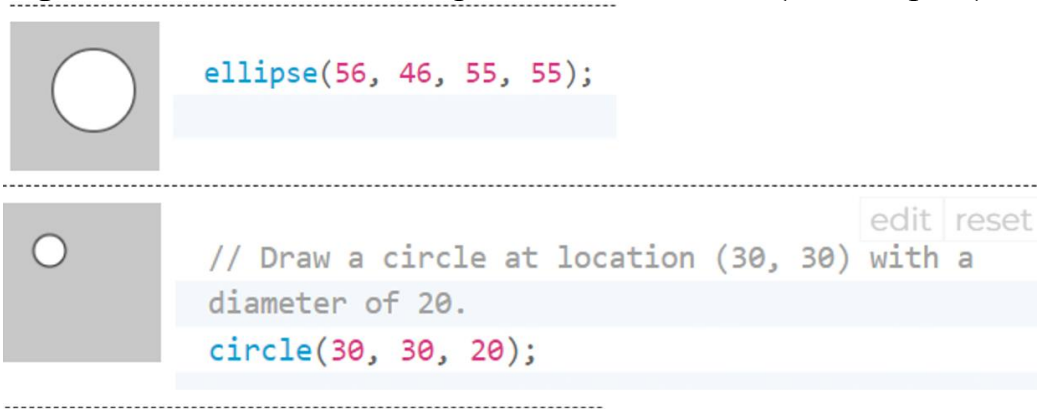


Figura 1.4. Exemple de program și rezultatul acestuia pentru funcțiile circle și ellipse

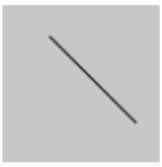
Funcția line(): trasează o linie dreaptă între două puncte pe ecran. Pentru a desena o linie de diferite grisimi, poate fi utilizat atributul stroke(). Linia nu poate fi umplută, deci atributul fill() nu va afecta aceasta. Liniile 2D sunt desenate implicit cu o lățime de un pixel, dacă dorim să specificăm grosimea liniei folosim funcția strokeWeight().

Sintaxa:

```
line(x1, y1, x2, y2)
line(x1, y1, z1, x2, y2, z2)
```

parametrii:
x1: x- coordonata primului punct;
y1: y- coordonata primului punct;
x2: x- coordonata punctului doi;

y2: y- coordonata punctului doi;
z1: coordonata z primului punct;
z2: coordonata z punctului doi;
Exemplu:



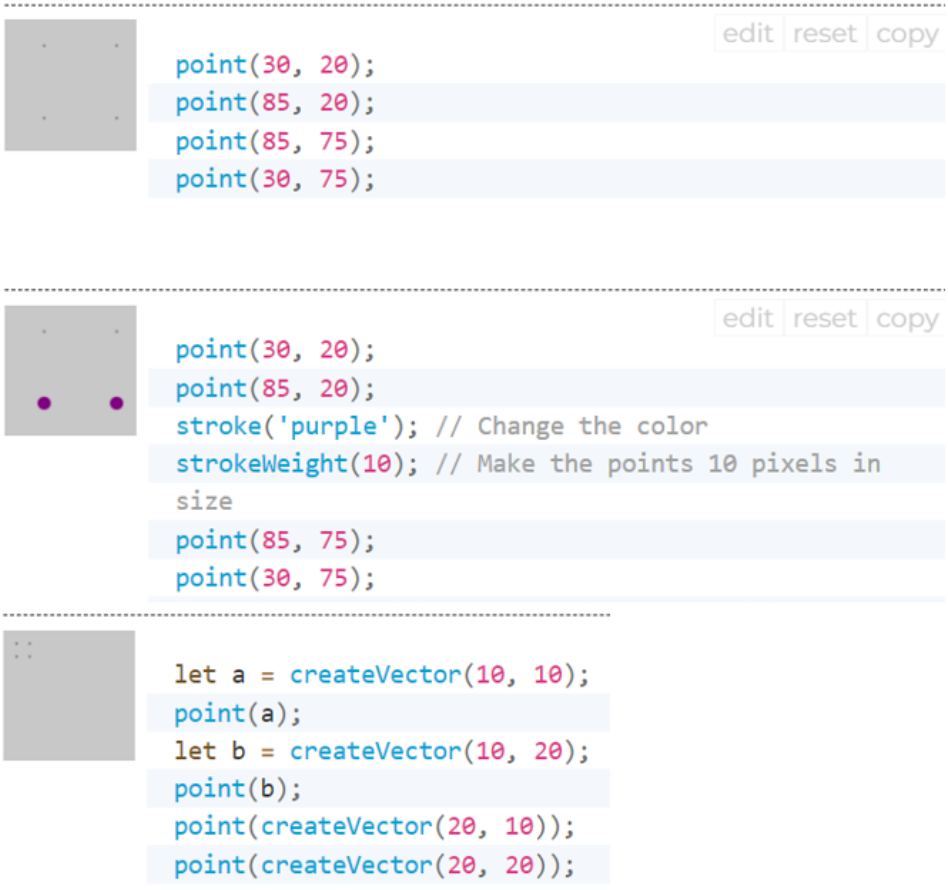
```
line(30, 20, 85, 75);
```

Funcția point(): desenează un punct, o coordonată în spațiu, cu dimensiunea de un pixel. Primul parametru este valoarea orizontală a punctului, al doilea este valoarea verticală a punctului. Culoarea punctului poate fi schimbată folosind funcția stroke(). Mărimea punctului se modificată folosind funcția strokeWeight().

Sintaxa:

```
point(x, y, [z])
point(coordinate_vector)
```

parametrii:
x: coordonata x;
y: coordonata y;
z: coordonata z (în regimul WebGL) (opțional);
coordonate_vector: un vector de coordonate;
Exemple de program și rezultatul acestuia pentru funcția point sunt reprezentate în figura 1.5



```
point(30, 20);
point(85, 20);
point(85, 75);
point(30, 75);
```

```
point(30, 20);
point(85, 20);
stroke('purple'); // Change the color
strokeWeight(10); // Make the points 10 pixels in
size
point(85, 75);
point(30, 75);
```

```
let a = createVector(10, 10);
point(a);
let b = createVector(10, 20);
point(b);
point(createVector(20, 10));
point(createVector(20, 20));
```

Figura 1.5. Exemple de program pentru funcția point

Funcția quad(): desenează un patrulater (un poligon cu patru laturi), unghiurile dintre laturi nu sunt limitate la nouăzeci de grade. Prima pereche de parametri (x1, y1) specifică primul vârf, iar perechile ulterioare trebuie să se miște în sensul acelor ceasornicului sau în sens invers acestora în jurul unei formei predefinite. Argumentele z funcționează numai când quad() este utilizat în modul WEBGL.

Sintaxa:

```
quad(x1, y1, x2, y2, x3, y3, x4, y4)
quad(x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4)
```

parametrii:
x1: x- coordonata primului punct;
y1: y- coordonata primului punct;
x2: x- coordonata punctului doi;
y2: y- coordonata punctului doi;
x3: x- coordonata punctului trei;
y3: y- coordonata punctului trei;
x4: x- coordonata punctului patru;
y4: y- coordonata punctului patru;
z1: coordonata z primului punct;
z2: coordonata z punctului doi;
z3: coordonata z punctului trei;
z4: coordonata z punctului patru;

Funcția rect (): desenează un dreptunghi pe ecran, o figură cu patru laturi cu fiecare colț de nouăzeci de grade. În mod implicit, primii doi parametri specifică poziția colțului din stânga sus, al treilea parametru specifică lățimea, iar al patrulea parametru specifică înălțimea. Cu toate acestea, modul în care acești parametri sunt interpretați poate fi modificat folosind funcția rectMode().

Al cincilea, al șaselea, al șaptelea și al optulea parametri, dacă sunt specificati, definesc raza pentru colțurile din stânga sus, din dreapta sus, din dreapta jos și, respectiv, din stânga jos. Parametrul razei colțului este setat la valoarea razei specificate anterior în lista de parametri.

Sintaxa:

```
rect(x, y, w, h, [tl], [tr], [br], [bl])
rect(x, y, w, h, [detailX], [detailY])
```

parametrii:
x: coordonata x a dreptunghiului;
y: coordonata y a dreptunghiului;
w : lățimea dreptunghiului;
h: înnălțimea dreptunghiului;
tl: raza colțului stânga-sus (opțional);
tr: raza colțului dreapta-sus (opțional);
br: raza colțului dreapta-jos (opțional);
bl: raza colțului stânga-jos (opțional);
detailX (număr întreg): numărul de segmente pe axa X (pentru regimul WebGL) (opțional);

detailY (număr întreg): numărul de segmente pe axa Y (pentru regimul WebGL) (opțional);

Exemple de program și rezultatele acestora sunt arătate în figura 1.6

Figura 1.6. Exemple de programuși pentru funcțiile rect și quad

Funcția square(): desenează un pătrat pe ecran, cu fiecare colț de nouăzeci de grade. Această funcție este un caz special al funcției rect() în care lățimea și înălțimea sunt aceleași, iar parametrul ***s*** pentru dimensiunea laturei. În mod implicit, primii doi parametri coordonata colțului din stânga sus, al treilea - dimensiunea laturii pătratului. Cu toate acestea, modul în care acești parametri sunt interpretați poate fi modificat folosind funcția rectMode().

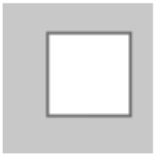
Al patrulea, al cincilea, al șaselea și al șaptelea parametrii, dacă sunt specificati, definesc raza pentru colțurile din stânga sus, din dreapta sus, din dreapta jos și, respectiv, din stânga jos. Parametrul razei colțului este setat la valoarea razei specificate anterior în lista de parametri.

Sintaxa:

```
square(x, y, s, [tl], [tr], [br], [bl])
```

parametrii:

x: coordonata x a pătratului;
y: coordonata y a pătratului;
s: mărimea laturei pătratului;
tl: raza colțului stânga-sus (opțional);
tr: raza colțului dreapta-sus (opțional);
br: raza colțului dreapta-jos (opțional);
bl: raza colțului stânga-jos (opțional);



edit reset copy

```
// Draw a square at location (30, 20) with a side
size of 55.
square(30, 20, 55);
```

edit reset copy

```
// Draw a square with rounded corners, each having
a radius of 20.
square(30, 20, 55, 20);
```

edit reset copy

```
// Draw a square with rounded corners having the
following radii:
// top-left = 20, top-right = 15, bottom-right =
10, bottom-left = 5.
square(30, 20, 55, 20, 15, 10, 5);
```

Figura 1.7. Exemplu de utilizare a funcției square

Funcția triangle(): desenează un triunghi. Argumenții funcției specifică coordonatele primul punct, coordonatele punctului doi, iar ultimele două argumente indică coordonatele punctului treilea al.

Sintaxa:

```
triangle(x1, y1, x2, y2, x3, y3)
```

parametrii:
x1: x- coordonata primului punct;
y1: y- coordonata primului punct;
x2: x- coordonata punctului doi;
y2: y- coordonata punctului doi;
x3: x- coordonata punctului trei;
y3: y- coordonata punctului trei;

```
triangle(30, 75, 58, 20, 86, 75);
```

Figura 1.8 Exemplu de utilizare a funcției triangle

1.2 Atribute grafice simple

Particularitățile figurilor geometrice, cum at fi culoarea figurii, grosimea și culoarea liniei tipul haşirării, modul de conexiune a liniilor reprezintă atributele grafice simple. În biblioteca p5.js sunt o listă de funcții care permit setarea sau modificarea acestor atribute.

Principalele atributele grafice sunt descrise în continuare:

Funcția fill(): Setează culoarea folosită pentru umplerea figurilor, toate figurile desenate după această funcție umplute (colorate) cu culoarea indicate ca parametru funcției. Această culoare este specificată în termeni de culoare RGB sau HSB, în funcție de colorMode() curent. (Spațiul de culoare implicit este RGB, fiecare valoare variază de la 0 la 255).

Sintaxa:

```
fill(v1, v2, v3, [alpha])
fill(value)
fill(gray, [alpha])
fill(values)
fill(color)
```

parametrii:

v1 (număr întreg):	roșu sau o valoare nuanței în raport cu gama de culori curentă;
v2 (număr întreg):	verde sau valoare saturației în raport cu gama de culori curentă;
v3 (număr întreg):	albastru sau valoarea luminozității în raport cu gama de culori curentă;
alpha (număr întreg):	(opțional);
value (un string):	string care indică culoarea;
gray (număr întreg):	valoarea culorii sure;
values (numere []):	un tabel care conține componentele culorilor:rușii, verde, albastră și alpha;

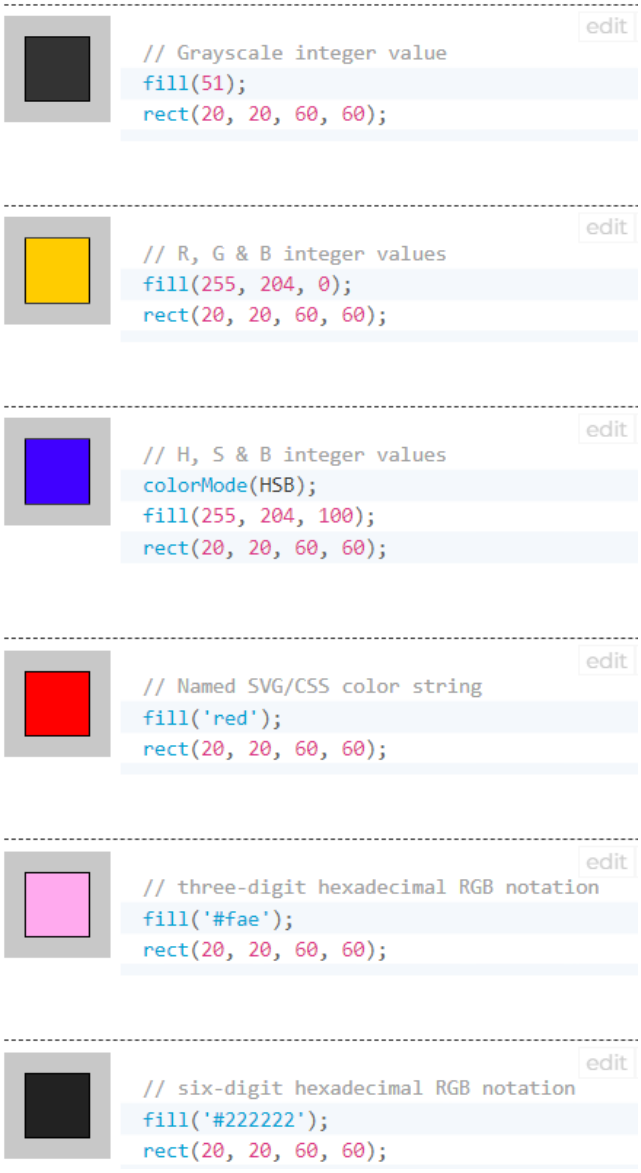


Figura 1.8 a) Exemple de utilizare a funcției fill

Dacă dorim ca figura să fie transparentă atunci utilizăm funcția **nofill**.

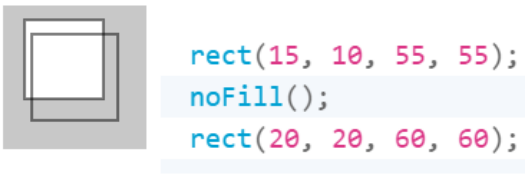


Figura 1.8 b) Exemple de utilizare a funcției nofill

Funcția stroke (): Specifică culoarea folosită pentru a desena liniile și marginile figurilor. Această culoare este specificată în termeni de culoare RGB sau HSB, în funcție de colorMode() curent (spațiul de culoare implicit este RGB, fiecare valoare variază de la 0 la 255). Intervalul alfa implicit este, de asemenea, de la 0 la 255.

Sintaxa:

```
stroke(v1, v2, v3, [alpha])
stroke(value)
stroke(gray, [alpha])
stroke(values)
```

`stroke(color)`
parametrii:

v1 (număr întreg): roșu sau o valoare nuanței în raport cu gama de culori curentă;
v2 (număr întreg): verde sau valoare saturației în raport cu gama de culori curentă;
v3 (număr întreg): albastru sau valoarea luminozității în raport cu gama de culori curentă;
alpha (număr întreg): (opțional)
value (un string): string care indică culoarea;
gray (număr întreg): valoarea culorii sure;
values (numere []): un tabel care conține componentele culorilor:rușii, verde, albastră și alpha;

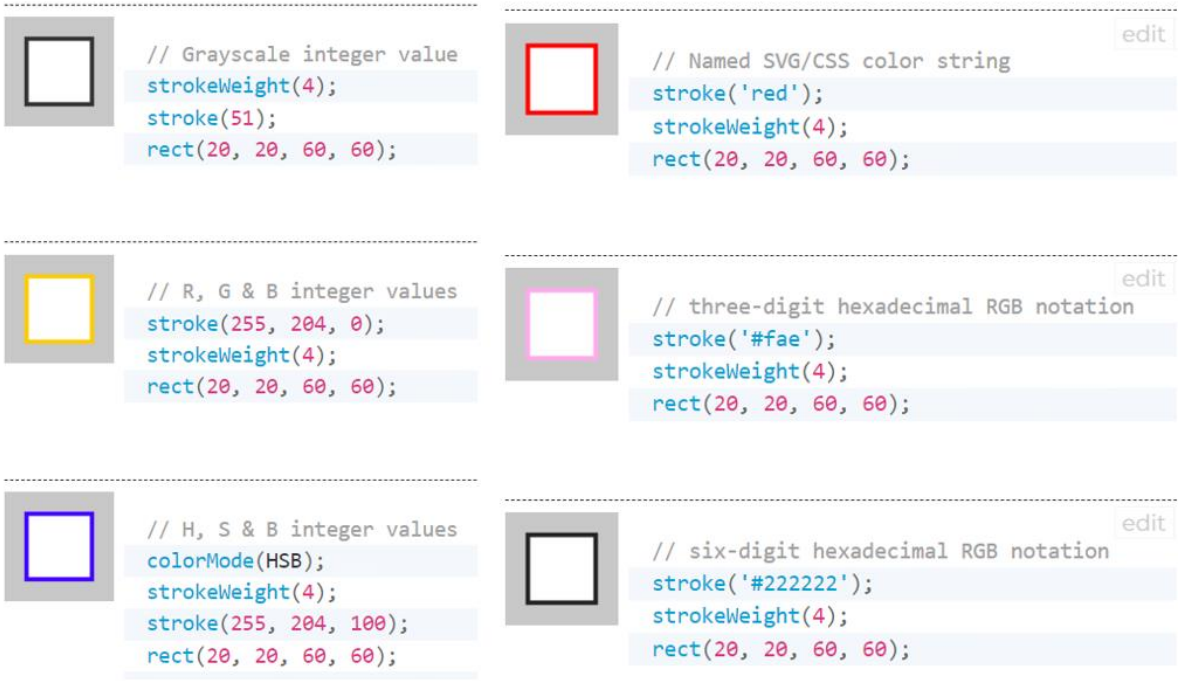


Figura 1.9 a). Exemple de utilizare a funcției stroke

Dacă dorim ca figura să fie desenată fără margine atunci utilizăm funcția **nostroke**.

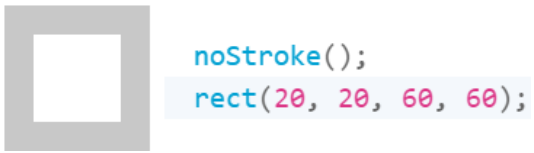


Figura 1.9 b). Exemple de utilizare a funcției nostroke

În afara culorii funcția stroke poate specifica următoatele:
`strokeCap()`
`strokeJoin()`
`strokeWeight()`

Funcția strokeWeight(number): Setează lățimea liniei. Toate valorile sunt specificate în pixeli.

Funcția strokeCap(cap): Setează stilul de redare a sfârșitului liniei. Aceste capete sunt fie rotunjite, pătrate sau extinse, fiecare dintre acestea fiind specificat cu parametrii corespunzători: ROUND, SQUARE și PROIECT. Implicit este ROUND.

Exemple de utilizare a acestor argumente sunt prezentate în figura 1. 10.

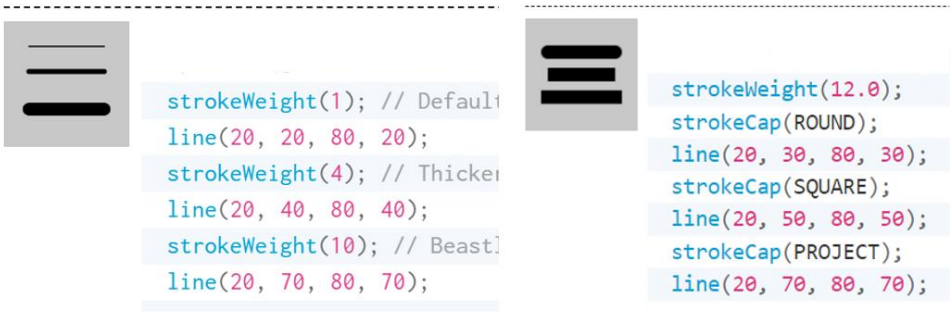


Figura 1.10. Exemple de utrilizare a funcției stroke

Funcția strokeJoin(join): Setează stilul conectării (unirii) segmentelor de linie. Aceste pot fi specificate cu parametrii corespunzători MITRE, BEVEL și ROUND. Implicit este setat ca MITRE.

Diferența dintre aceste moduri este prezentată în figura 1.11.

Figura 1.11 Parametrii funcției strokeJoin

În cazul în care este nevoie de adăugat text pot fi utilizată funcția text care poate avea diferiți parametrii.

Funcția text(): Desenează text pe ecran, afișează informațiile specificate în primul parametru pe ecran în poziția specificată de parametrii suplimentari. Se va folosi un font implicit, cu excepția cazului în care un font este setat cu funcția **textFont()** și o dimensiune implicită va fi folosită dacă nu este setat un font cu **textSize()**. Culoarea textului poate fi schimbată cu funcția **fill()**. Schimbareea conturului textului se efectuează cu funcțiile **stroke()** și **strokeWeight()**.

Textul este afișat în relație cu funcția **textAlign()**, care oferă opțiunea de a desena la stânga, la dreapta și în centrul coordonatelor.

Sintaxa:

```
text(str, x, y, [x2], [y2])
```

Parametrii:

str : se indică șirul pentru afișare pe ecran;

x: coordonata x a punctului de început a textului;

y: coordonata y a punctului de început a textului;

x2 și y2: definesc o zonă dreptunghiulară în care să se afișeze și pot fi utilizați numai cu date de tip șir de caractere. Când acești parametri sunt specificați, ei sunt interpretați pe baza setării curente rectMode(). Textul care nu se încadrează complet în dreptunghiul specificat nu va fi desenat pe ecran. Dacă x2 și y2 nu sunt specificate, alinierea liniei de bază este implicită, ceea ce înseamnă că textul va fi desenat în sus de la x și y.

Principalele atributele textului sunt aduse în figura 1.12

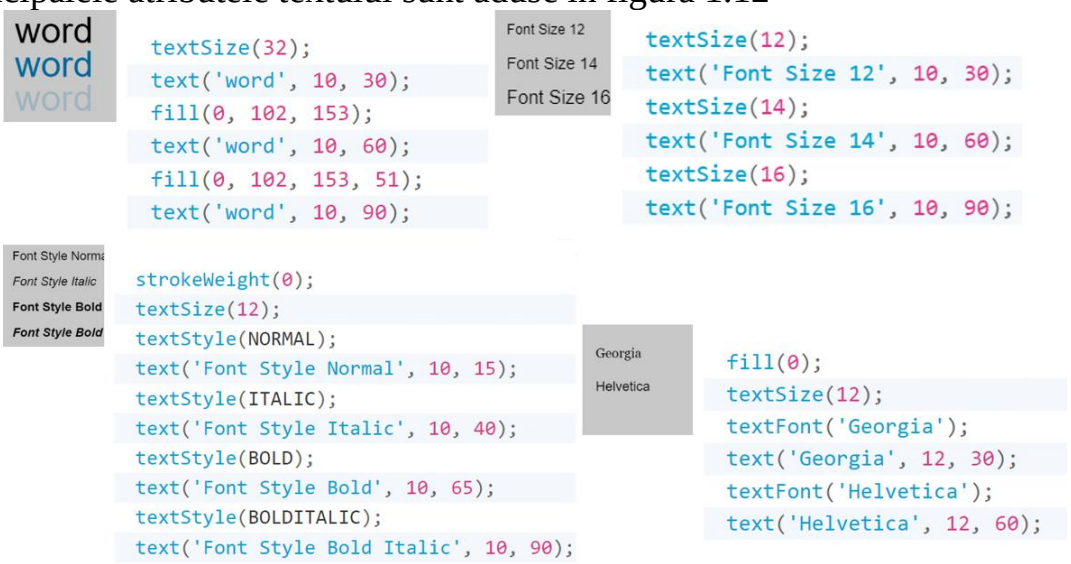


Figura 1.12 Exemple de utilizare a funcțiilor de lucru cu textul

1.3 Formatele grafice

Orice imagine grafică este salvată în fișier. Modul în care datele grafice sunt stocate într-un fișier determină formatul grafic al fișierului.

Format- structura fișierului, care determină modul în care sunt stocate și afișate pe ecran sau la imprimare datele. Formatul fișierului este de obicei indicat în numele său, ca o parte separată printr-un punct (de obicei, această parte se numește extensia numelui fișierului).

Compresia este utilizată pentru fișierele grafice, deoarece au un volum destul de mare de informație pe care o conțin, fiecare format are propriul algoritm prin care sunt comprimate datele conținute în fișier.

Formatele grafice se clasifică după:

- - tipul datelor stocate (raster, vector și forme mixte),
- - în funcție de cantitatea admisă de date
- - parametrii imaginii
- - modul de stocare a paletei de culori
- - metoda de compresie a datelor
- - prin metode de organizare a fișierelor (text, binar)

Din varietatea de formate, nu există niciunul ideal care să satisfacă toate cerințele posibile. Alegerea unuia sau a altui format pentru salvarea unei imagini depinde de obiectivele și scopuri de lucru cu imaginea. Dacă aveți nevoie de o acuratețe fotografică a culorilor, atunci este preferat unul dintre formatele raster. Pentru sigle, diagrame, elemente de design sunt recomandabile formate de stocare vectoriale. Formatul fișierului afectează cantitatea de memorie pe care o ocupă fișierul. Editorii grafici permit utilizatorului să aleagă independent formatul pentru salvarea imaginii. Există formate de fișiere grafice universale care acceptă atât imagini vectoriale, cât și imagini bitmap în același timp.

În tabelul 1.1 este arătată o scurtă descriere a formatelor fișierelor grafice utilizate.

Tabelul 1.1. Caracteristicile formatelor grafice

Format	Mod imagine	Tipul informațiilor grafice	Cerere
BMP	Numai culorile indexate	Modele de tipul aplicațiilor care conțin zone întinse de culoare solidă	Formatul este acceptat de toate aplicațiile. Nu este utilizat în publicare din cauza volumului mare de fișiere
Tiff	Tot	Imagini de tip diagramă	Un format considerat formatul preferat pentru realizarea de machete axate pe tipărirea tipografică și alte metode de reproducere
PSD	Suportă toate tipurile de imagini	Orice imagini	Intern pentru program Adobe Photoshop. Singurul format în care sunt salvate toate informațiile despre un document, inclusiv straturile și canalele. Cu toate acestea, este mai bine să salvați imaginea terminată în alte formate grafice din două motive. La început, Fișier PSD cu dimensiuni mult mai mari. În al doilea rând, acest format nu este importat de aspectul și programele de grafică obiect.

Tabelul 1.1. (continuare)

Format	Mod imagine	Tipul informațiilor grafice	Cerere
Jpeg	Imagini color complet doar la	Fotografii depline sau mostre de grafică artistică,	Conceput pentru a salva fișiere punct cu compresie. Comprimarea utilizând această metodă reduce dimensiunea fișierului de la zecimi la sută la o sută

	modelele RGB și CMYK	inclusiv revărsări subtile de culori.	de ori (interval practic - de la 5 la 15 ori), dar compresia în acest format are pierderi (în limite acceptabile). Un algoritm de compresie foarte eficient a condus la cea mai largă distribuție a JPEG pe World Wide Web. Nu este recomandată utilizarea acestui format în industria tipografică.
GIF	Numai imagini indexate	Desene de tip diagramă - imaginile au suprafețe mari de culoare uniformă cu limite bine definite; imagini animate	Proiectat special pentru transmiterea imaginilor în rețelele globale. Are cea mai eficientă metodă de compresie, care este necesară pentru a reduce timpul de transmisie a imaginilor. O nouă versiune permite stocarea mai multor imagini într-un singur fișier. Cea mai obișnuită utilizare a acestei caracteristici este pe web. Browserul web afișează imagini situate în Fișier GIF, secvențial.
Imagine PNG	Suportă imagini color RGB și imagini indexate	Imagini color cu tranziții fluide de la zone opace la zone transparente	Numele formatului, Portable Network Graphics, vorbește despre scopul său - de a transfera imagini pe rețele. Este posibil să utilizați un singur canal suplimentar pentru stocarea măștii de transparență. Are un algoritm de compresie eficient, fără pierderi de informații. Formatul este utilizat pe web
EPS	Tot	Grafică vectorială, fonturi, imagini rasterizate	Este utilizat în industria tipografică. Este posibil să stocați informații despre rasterizare, contururi și curbe de calibrare.

Surse bibliografice:

- https://p5js.org/reference/
- https://github.com/processing/p5.js/wiki/p5.js-overview
- ”Make: Getting started with p5.js” Lauren McCarthy, Casey Reas, Ben Fry;
- ”Learn JavaScript with p5.js” Engin Arslan

Lucrarea de laborator 1
Tema: Studierea primitivelor grafice simple 2D

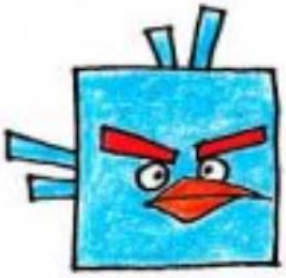
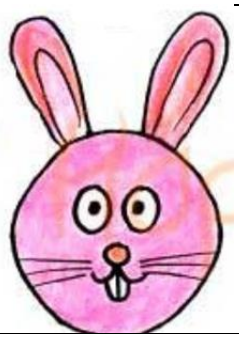
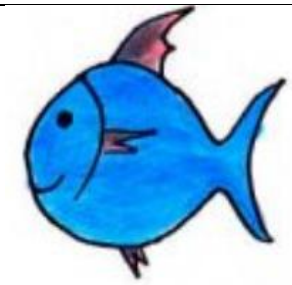
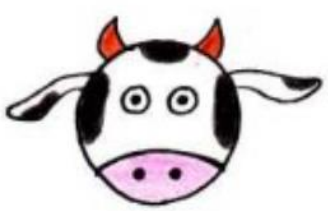
Scopul lucrării: Obținerea cunoștințelor practice în sinteza scenelor grafice 2D statice, utilizând primitivele grafice simple a bibliotecii p5.js.

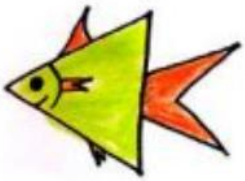
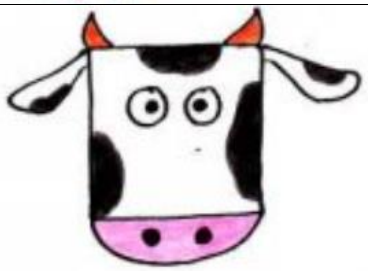
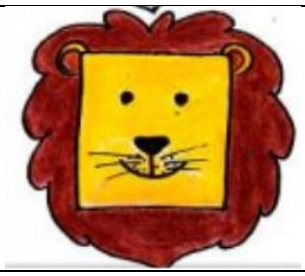
Sarcina lucrării:

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

2. Elaborați un program care creează personajul conform variantei https://ja-rastu.ru/raskraski/uroki/1079-poshagovoe-risovanie-zhivotnyh-iz-geometricheskih-figur.htmlindicate de profesor. Variantele sunt indicate în tabelul 1.2. Pentru crearea acestei imagini pe pași puteți consulta pagina

Tabelul 1.2 Variantele pentru realizarea lucrarii de laborator

Varianta	Personajul	Varianta	Personajul
1		11	
2		12	
3		13	
4		14	
5		15	

6		16	
7		17	
8		18	
9		19	
10		20	

Exemplu de program realizat în p5.js:

```
var Width;
var Height;
var CurrentY;

function setup() {
  Width = 400;
  Height = 734;
  createCanvas(Width, Height);
}

function draw() {
  background(220);
  CurrentY = Height - 20;

  //Realizam baza (1 parte)
  stroke(6, 21, 131);
  for (let i = 0; i < 20; i++) {
    line(0 + 20, CurrentY, Width - 20, CurrentY);
```

```
  CurrentY--;
}

// Realizam baza (2 parte)
stroke(15, 23, 71);
for (let i = 0; i < 20; i++) {
  line(0 + 20 + i, CurrentY, Width - 20 - i, CurrentY);
  CurrentY--;
}

//Основание двери
CurrentY += 10;
stroke(6, 21, 121);
for (let i = 0; i < 550; i++) {
  line(0 + 40, CurrentY, Width - 40, CurrentY);
  CurrentY--;
}

//Колонки
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);

//Дверные ямы
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);
  }
}

//Средняя колонка
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);

//Дверь
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);

//Окна
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);
}

//Верхушка
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:

Întrebări de control:

1. Numiți primitive grafice simple.
2. Cum poate fi realizată modificarea atributelor de afișare ale primitivelor grafice?
3. Cum poate fi scris textul în mod grafic?
4. Numiți formate standard pentru imagini.