

BARBULESCU

1. Care dintre urmatoarele valori reprezinta culoarea **rosu** in modelul de culoare RGB?
a) #0000FF
b) #00FF00
c) #FF0000 -> (255,0,0)
d) #000000

2. Care dintre urmatoarele valori reprezinta culoarea **albastru** in modelul de culoare RGB?
a) #000000
b) #FF0000
c) #00FF00
d) #0000FF -> (0,0,255)

3. Care dintre urmatorii selectori CSS selecteaza elementul cu id-ul „maria” din document?
a) .maria
b) body[maria]
c) maria
d) #maria

4. Care este apelul corect pentru preluarea intr-un obiect a canvas-ului?
a) var b = canvas.getElementById("test")
b) var b = document.getElementById("test")
c) var b = document.getcanvas("test")
d) var b = document.getcontext("test")

5. Formatul de stocare SVG este specific stocarii:
a) fisierelor de tip sunet
b) imaginilor necomprimate
c) imaginilor vectoriale
d) imaginilor de tip raster

6. In CSS prin table#unu se selecteaza:
a) Prima linie din table
b) Tabelul cu id=unu
c) Tabelul cu o linie
d) Un buton

7. Formatul de stocare a imaginii JPEG este specific:
a) imaginilor vectoriale
b) imaginilor de tip raster

- c) fisierelor de tip sunet si video
 - d) imaginilor necomprimate
8. Desenarea unui cerc folosind un context grafic asociat unui element canvas se realizeaza prin intermediul functiei:
- a) closedPath
 - b) circle
 - c) ellipse
 - d) arc**
9. Elementul HTML utilizat pentru introducerea unei imagini in cadrul unei pagini este:
- a) image
 - b) class
 - c) img**
 - d) graph
10. Care dintre urmatoarele valori reprezinta culoarea verde in modelul de culoare RGB?
- a) #000000
 - b) #0000FF
 - c) #00FF00 -> (0,255,0)**
 - d) #FF0000
11. Care dintre urmatorii selectori CSS selecteaza elementul cu id-ul „ion” din document?
- a) body[ion]
 - b) #ion**
 - c) ion
 - d) .ion
12. Ce element HTML este utilizat pentru introducerea unei secvente CSS in interiorul unui document?
- a) link
 - b) class
 - c) css
 - d) style**
13. Grafica vectoriala foloseste o reprezentare sub forma de:
- a) functii matematice**
 - b) matrice de puncte
 - c) transformari Fourier
 - d) coeficienti polinomiali

14. Care dintre urmatoarele modele de culoare este de tip aditiv?

- a) CMYK
- b) RGB
- c) HSL
- d) HSV

15. Grafica raster (bitmap) foloseste o reprezentare sub forma de:

- a) matrice de puncte
- b) coeficienti polinomiali
- c) functii matematice
- d) transformari Fourier

16. Elementul HTML folosit pentru reprezentarea unui paragraf este:

- a) paragraph
- b) li
- c) section
- d) p

AU RASPUNSURILE DE PE PLATFORMA!

1. Care dintre urmatoarele NU sunt formate de stocare pentru imagini vectoriale?
a) PNG
b) SHP
c) DXF
d) EPS
2. Metoda corecta pentru desenarea unui text pe un element de tip canvas este:
a) appendText(text, x, y);
b) writeText(text, x, y);
c) drawText(text, x, y);
d) fillText(text, x, y);
3. Care dintre urmatoarele atribute nu este utilizat in mod obisnuit pe un element audio?
a) loop
b) showControls
c) volume
d) autoplay
5. CMYK este un model:
a) aditiv
b) subtractiv
c) nu este un model utilizat in multimedia
d) multiplicativ
6. Care dintre urmatoarele nu este o metoda de transformare disponibila pentru CanvasRenderingContext2D(canvas.getContext("2d")):
a) rotatie
b) translatie
c) deplasare
d) scalare

4. Care dintre urmatoarele raspunsuri reflectă conținutul vectorului data, obținut după cum urmează, în cazul imaginii de mai jos.

```
const context = canvas.getContext("2d")
const imageData = context.getImageData(0,0,2,2)
const data = imageData.data;
```

4 întrebare

Corect

Marcat 1,00 din 1,00

▼ Întrebare cu flag

Care dintre următoarele răspunsuri reflectă conținutul vectorului data, obținut după cum urmează, în cazul imaginii de mai jos.

```
const context = canvas.getContext("2d")
const imageData =
context.getImageData(0,0,2,2)
const data = imageData.data;
```

Alegeți o opțiune:

- a. [255,255,255,255, 197,52,144,255, 255,0,0,255, 0,0,255]
- b. [255,0,0,255, 255,255,255,255, 0,0,255, 197,52,144,255]
- c. [255,0,0,255, 0,0,0,255, 255,255,255, 197,52,144,255]
- d. [255,0,0,255, 0,0,0,255, 255,255,255, 197,52,144,255]

The correct answers are: [255,0,0,255, 0,0,0,255, 255,255,255, 197,52,144,255], [255,0,0,255, 0,0,0,255, 255,255,255, 197,52,144,255]

5 întrebare

7. Care dintre urmatoarele afirmații nu este adevarata în cazul graficii vectoriale:

- a) contin forme geometrice precum puncte, linii, curbe etc.
- b) un set de comenzi este folosit pentru a desena imaginea
- c) calitatea lor este afectată atunci cand sunt scalate**
- d) pentru imagini simple, acestea ...

8. Urmatoarele proprietăți sunt disponibile pentru CanvasRenderingContext2D(canvas.getContext("2d")):

- a) textAlign**
- b) strokeStyle**
- c) font**
- d) lineWidth**

9. Desenarea conturului unui dreptunghi pe un element de tip <canvas> se poate realiza cu urmatoarele metode aferente CanvasRenderingContext2D:

- a) `paintRect(x, y, width, height)`
- b) `strokeRect(x, y, width, height)`**
- c) `drawRect(x, y, width, height)`
- d) `rect(x, y, width, height)`

10. Care dintre urmatoarele metode nu există implicit în JavaScript?

- a) `document.getElementsByName();`
 - b) `document.getElementByTagName();`**
 - c) `document.getElementsByTagName();`
 - d) `document.getElementById();`
- !!! DAR există `document.getElementsByTagName()`**

CELE DE MAI SUS AVEAU RASPUNSURILE DE PE PLATFORMA!

IONITA

The screenshot shows a browser window with a code editor on the left and a canvas on the right. The code editor contains the following JavaScript code:

```
File aplicatie.html
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    let canvas, context, x = 100, y = 100, w = 40, h = 40;

    function desenare() {
        context.fillStyle = 'silver';
        context.fillRect(0, 0, 500, 500);
        context.fillStyle = 'red';
        context.fillRect(x, y, w, h);
        requestAnimationFrame(desenare);
    }

    function aplicatie() {
        canvas = document.querySelector('canvas');
        context = canvas.getContext('2d');
        desenare();
        setInterval(() => { y += 2; }, 50);
    }
</script>
</head>
<body><canvas width="500" height="500"></canvas></body></html>
```

The canvas element has a width of 500 and a height of 500. A red square is drawn at position (100, 100) with a size of 40x40 pixels. The code uses a recursive call to `requestAnimationFrame` to continuously draw the red square at a y-position that increases by 2 pixels every 50ms.

Patratul rosu se va deplasa:

- a) in sus cu o viteza de 100 pixeli/secunda
- b) in jos cu o viteza de 50 pixeli/secunda
- c) in jos cu o viteza de 100 pixeli/secunda
- d) **in jos cu o viteza de 40 pixeli/secunda**
- e) in sus cu o viteza de 50 pixeli/secunda

Care dintre urmatoarele valori reprezinta culoarea rosu in modelul de culoare HSL?

- a) (0, 100%, 0%)
- b) (120, 100%, 100%)
- c) **(0, 100%, 50%)**
- d) (120, 100%, 0%)
- e) (240, 0%, 0%)

Fie **a** o referinta la un element HTML de tip audio. Pozitionarea la jumatarea secenteii audio se realizeaza prin:

- a) a.time = a.duration * 0.5
- b) a.currentTime = 0.5
- c) a.time = a.duration ? a.duration * 0.5 : 0
- d) **a.currentTime = a.duration ? a.duration / 2 : 0** ????????????????
- e) a.position = 1 / 2

C este o referinta la un canvas de latime 20 pixeli si inaltime 10 pixeli.

Fie codul:

```
let context = C.getContext("2d");
let imageData = context.getImageData(0, 0, W, H);
```

Componenta de culoare verde pentru pixelul de la linia 3, coloana 4 este accesata prin (numararea liniilor si coloanelor incepe de la 0):

- a) imageData.data[80]
- b) imageData[97]
- c) **imageData.data[81]**
- d) imageData[81]
- e) imageData.data[97]

Fie documentul HTML:

```
<!DOCTYPE html>

<html>

<head></head>

<body>

<svg>

<rect id="r1" x="10" y="10" width="10" height="10"></rect>

<rect id="r2" x="10" y="10" width="10" height="10"></rect>

</svg>

</body></html>
```

Pozitionarea dreptunghiului **r2** imediat sub dreptunghiul **r1** se poate realiza prin urmatoarea secventa JavaScript:

- a) `document.querySelector('#r2').setAttribute('y', '10');`
- b) `document.querySelector('#r1').y = 20;`
- c) `document.querySelector('r2').setAttribute('y', '20');`
- d) `document.querySelector('#r1').setAttribute('x', '20');`
- e) `document.querySelector('#r2').setAttribute('y', '20');`

GRILE ADUNATE

1. Care dintre algoritmii de mai jos este doar fara pierdere de informatie?
 - a) **FLAC**
 - b) Audio Mpeg Layer 3
 - c) WMA
 - d) Audio Mpeg-4

2. Care din cele de mai jos NU este cerinta pentru compresia video
 - a) Acces random la sechete
 - b) Posibilitate editare
 - c) Corectie/evitare erori
 - d) **Generarea cadrelor intermediare**

3. Care din cele de mai jos NU este cerinta pentru compresia video
 - a) Reverse playback
 - b) **Compresia sunetului fara pierdere de informatie**
 - c) Sincronizare audio video
 - d) Viteza mare codare/decodare

4. Compresia video are la baza eliminarea redundantei:
 - a) doar audio
 - b) doar inter-cadru
 - c) **intra si inter-cadru**
 - d) doar intra-cadru

5. In cadrul compresiei JPEG pentru imagini este utilizata:
 - a) Transformata Fourier Discreta
 - b) Compresia Lempel-Ziv-Welch
 - c) **Transformata Cosinus Discreta**
 - d) Transformata Sinus Discreta

6. Care este atributul care face ca sunetul in tag-ul audio sa ruleze continuu pana cand este oprit de catre utilizator?
 - a) controls
 - b) **loop**

7. Care este elementul introdus in html5 care permite redarea nativa a fisierelor de sunet?
- a) <sound>
 - b) <audio> ?
 - c) <video>
 - d) <play>
8. Care informatie NU este adevarata cu privire la Group of Pictures (GOP)
- a) Sunt secvențe repetitive I,P,B
 - b) Reprezinta cadre in ordinea aparitiei
 - c) Nu sunt comprimate
 - d) Incep mereu cu care I
9. Pentru ce se utilizeaza CSS in contextul HTML5?
- a) Color style sheets
 - b) Clear style sheets
 - c) Computer style sheets
 - d) Cascading style sheets
10. Care dintre urmatorii selectori CSS selecteaza elementul cu id-ul „ion” din document?
- a) body[ion]
 - b) #ion
 - c) ion
 - d) .ion
11. Care dintre urmatoarele formate este utilizat pentru stocarea sunetului necomprimat?
- a) JPG
 - b) WAVE
 - c) MP3
 - d) BMP
12. Sunetul este definit ca:
- a) energie magnetica intr-un mediu elastic
 - b) o vibratie care se propaga printre un mediu material
 - c) energie electrica propagata prin vid
 - d) energie electrica statica transmisa prin orice mediu

13. Ce atribut este utilizat pentru specificarea stilurilor CSS intr-un element HTML?

- a) css
- b) link
- c) **style**
- d) class

14. Care este variabila corecta de includere a imaginii ase.jpg in cadrul unui document HTML?

- a) <image src="ase.jpg" alt="Exemplu imagine">
- b) ase.jpg
- c) ****
- d)

((Care este VARIANTA corecta de includere a imaginii ase.jpg in cadrul unui document HTML?))

15. Elementul HTML folosit pentru reprezentarea unui rand in cadrul unui tabel este:

- a) **tr**
- b) table
- c) cell ---- mai e o varianta cu row aici
- d) td
- e) tc

<tr> de la table row

!!! <td> = celula cu date, celula standard si <th> este celula din header

((Elementul HTML folosit pentru reprezentarea unei linii in cadrul unui tabel este:))

16. Elementul HTML utilizat pentru introducerea unui element de grafica **raster** in cadrul unei pagini este:

- a) graph
- b) svg
- c) **canvas**
- d) vector

17. Stabilirea culorii rosu pentru **umplerea** urmatoarei **figuri desenate** pe contextul grafic c asociat unui element canvas se realizeaza prin:

- a) `c.strokeStyle("red")`
- b) `c.fillStyle = "red"`
- c) `c.color = "red"`
- d) `c.strokeStyle = "red"`

18. Stabilirea culorii rosu pentru **desenarea** urmatoarei **linii** pe contextul grafic c asociat unui element canvas se realizeaza prin:

- a) `c.strokeStyle("red")`
- b) `c.fillStyle = "red"`
- c) `c.color = "red"`
- d) `c.strokeStyle = "red"`

19. Care dintre urmatoarele metode NU exista implicit in JavaScript?

- a) `document.getElementsByName()`
- b) `document.getElementByName()`**
- c) `document.getElementsByTagName()`
- d) `document.getElementById()`

!!! DAR exista `document.getElementsByTagName()`

20. Se creaza jQuery ~~`$(“p.class1, #p1”).css(“border”, “solid green”);`~~ modifica culoarea chenarului pentru:

- a) toate elementele de tip paragraf (`p`) din pagina care au id-ul `#p1` si clasa CSS `class1`
- b) selectorul jQuery este incorrect
- c) toate elementele de tip paragraf (`p`) din pagina care au fie id-ul `#p1`, fie clasa CSS
- d) toate elementele din pagina de tip paragraf (`p`) avand clasa CSS `class1` si elementul din pagina cu id-ul `p1`**

21. Echivalentul jQuery pentru: `document.getElementById(‘elementId’)` este:

- a) `$("elementId");`
- b) `$(".elementId");`
- c) `$("#elementId");`**
- d) `$("<elementId>");`

22. jQuery este:

- a) Un limbaj de programare
- b) Un limbaj de acces la baza de date
- c) O biblioteca JavaScript**
- d) Un tag

23. Desenarea unui dreptunghi în canvas se realizează cu metoda:

- a) `paint(x,y,a,b)`
- b) `fillRect(0,0,50,70)`
- c) `fillStyle(0,0,10,30)`
- d) `square(x,y,a,b)`

24. Care este apelul corect pentru preluarea unui obiect din canvas-ului?

- a) `var b = canvas.getElementById("test")`
- b) `var b = document.getElementById("test")`
- c) `var b = document.getcanvas("test")`
- d) `var b = document.getContext("test")`

25. Care este sintaxa corecta pentru introducerea in pagina a unui script extern?

- a) `<src type="text/javascript" src="scriptulmeu.js">`
- b) `<script type="text/javascript" src="scriptulmeu.js">`
- c) `<script type="text/javascript" href="scriptulmeu.js">`
- d) `<script type="text/javascript" name="scriptulmeu.js">`

26. Care este tagul pentru introducerea in HTML a unui container pentru crearea de grafica?

- a) `<paint>`
- b) `<style>`
- c) `<canvas>`
- d) `<graphics>`

((Care este tagul pentru introducerea in HTML a unui container pentru crearea grafica?))

27. Care este metoda corecta pentru scriere in canvas?

- a) `font(x,y,text)`
- b) `fillText(text,x,y)`
- c) `fillstyle(text,x,y)`
- d) `paint(text,x,y)`

28. Formatul de stocare SVG este specific stocarii:

- a) fisierelor de tip sunet
- b) imaginilor necomprimate
- c) `imaginilor vectoriale`
- d) imaginilor de tip raster

29. Compresia video are la baza eliminarea redundantei:

- a) doar audio
- b) doar inter-cadru
- c) intra si inter-cadru
- d) doar intra-cadru

30. Sunetul este definit ca:

- a) energie magnetica intr-un mediu elastic
- b) o vibratie care se propaga prin un mediu material
- c) energie electrica propagata prin vid
- d) energie electrica statica transmisa prin orice mediu

31. Compresia JPEG NU utilizeaza:

- a) compresie RLE
- b) compresia Huffman
- c) transformata cosinus discreta
- d) compresia LZW

32. Care dintre urmatoarele caracteristici NU este specifica imaginilor vectoriale:

- a) mentine semnatica imaginii
- b) obiectele componente sunt descrise matematic
- c) fisierul imagine este mic
- d) imaginea este dependenta de scara de vizualizare

33. Elementul HTML utilizat pentru introducerea unui element de grafica **vectoriala** in cadrul unei pagini este:

- a) graph
- b) svg
- c) canvas
- d) vector

!!! <canvas> pentru grafica raster

34. Care este varianta corecta pentru introducerea in HTML a unui CSS extern?

- a) <link rel="stylesheet" type="text/css" href="stilulmeu.css">
- b) <style src="stilulmeu.css" />
- c) <stylesheet>stilulmeu.css</stylesheet/>
- d) <script style="stilulmeu.css">

Care este varianta corecta pentru introducerea in HTML a unui CSS **extern?**

- a) <**link** rel="stylesheet" type="text/css" href="stilulmeu.css">

!!!DAR daca este **intern avem <style src="...">**

35. Pierderea de informatie in cazul compresiei JPEG este influentata de:

- a) **alegerea matricei de cuantizare**
- b) calitatea imaginii sursa
- c) parametrii compresiei Huffman
- d) parametrii compresiei RLE
- e) modalitatea de aplicare a transformatiei cosinus discreta

36. Ce element HTML este utilizat pentru specificarea stilurilor CSS externe pentru un document?

- a) css
- b) link**
- c) style
- d) class

1. Grafica vectoriala foloseste o reprezentare sub forma de: => functii matematice
2. Compresia MPEG 1-2 LAYER III foloseste codificarea perceptuala si un model psihacustic
3. GIF este un format specific => imaginilor de tip raster
4. Care dintre urmatoarele este un format de stocare pentru imagini raster? => JPG
5. Stabilirea culorii pentru culoarea urmatoarelor linii din contextul grafic:
⇒ context.fillStyle="red"

37. Care dintre urmatoarele sevante respecta sintaxe CSS?

- a) body:color=black
- b) **body(color:black)**
- c) (body, color.black)
- d) (body.color=black(body))

38. Ce elemente HTML vor fi modificate prin intermediul selectorului img.all?

- a) Toate imaginile din cadrul documentului
- b) toate elementele care au atributul id="img.all"
- c) toate elementele care au atributul class="img.all"
- d) **toate imaginile care au atributul class="all"**

39. Care din urmatoarele elemente svg este utilizat pentru gruparea unui set de elemente fara a le afisa?

- a)rect
- b)defs
- c)**g**
- d) svg

40. Care dintre urmatoarele valori nu este o valoare acceptabila pentru proprietatea CSS position?

- a) static
- b) relative
- c) fixed
- d) absolute
- e) **float**

Which is the HTML element used to draw a raster graphic on a web page? => **canvas**

Which of the following tags does include the “stud.jpg” image, in an HTML document? =>

Which of the following CSS selectors refers an element with “my” id, in an HTML document? => **#my**

Which colour model does use a double-cone representation for its colour space? => **HLS**

Which is the output of the following chunk of HTML & CSS code?

(ceva cu span)

What is the output of the following chunk of HTML & CSS code?

```
<div id="test">
  <span>Text</span>
</div>
<style>
div#test span { color: green; }
div span { color: blue; }
span { color: red; }
</style>
```

⇒ It colours the text in green.

Check all arguments accepted by the DrawImage JavaScript function:

⇒ a canvas element, an image, a video element

Check all the processing operations involved in JPEG compression algorithm:

⇒ transformare cosinus discreta, cuantizare, block splitting

Select ALL statements depicting DOM, from the following:

- ⇒ DOM is the acronym for Document Object Model
- ⇒ DOM provides the way to programmatically access HTML structure in JavaScript
- ⇒ HTML of every web page is turned into a DOM representation by the browser

Select all characteristics of a raster image, from the following

⇒ The raster image is figured as a matrix of points.

- ⇒ Raster image file size depends on the image size and the colour depth

Select ALL solutions for incorporating CSS in HTML

- ⇒ ?defined within a style block, in the head section of a web page
⇒ (tag link, tag style (in head), atribut style pt inline (in body))
⇒ c) este ambiguu -> Coded in the body of the web page

Select ALL correct statements referring to the FlexBox model

Select ALL correct statements referring to the FlexBox model.

Alegeți una sau mai multe opțiuni:

- a. It uses grid-column and grid-row properties to change the size of items.
- b. It is activated using display property
- c. It is used to change the horizontal or vertical layout of HTML elements, in a web page
- d. It can be used to programmatically change the order of the HTML elements.
- e. It is appropriate for creating a responsive web application

b, d, e, c? e ambiguu

Select ALL primary colours of the subtractive colour model.

cyan, magenta, yellow

What purpose has setTimeout JavaScript function?

Răspuns:

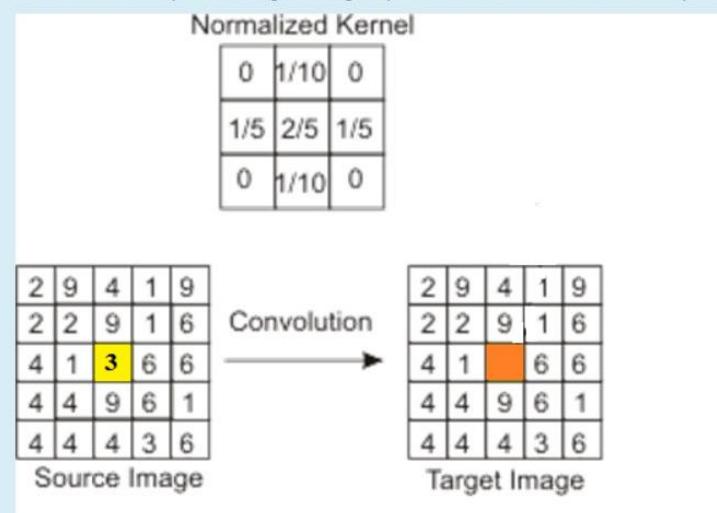
metoda setTimeout apeleaza o functie sau evalueaza o expresie dupa un numar specificat de milisecunde. 1000ms=1s si functia ce va fi executata va fi rulata o singura data. Pt a repeta executia functiei putem folosi metoda setInterval. Pt a preveni functia din a rula putem folosi metoda clearTimeout

setTimeout(function(){ alert("Hello"); }, 3000); - afiseaza o fereastra de alerta cu textul hello dupa 3 sec

What is HyperVideo?

Răspuns:

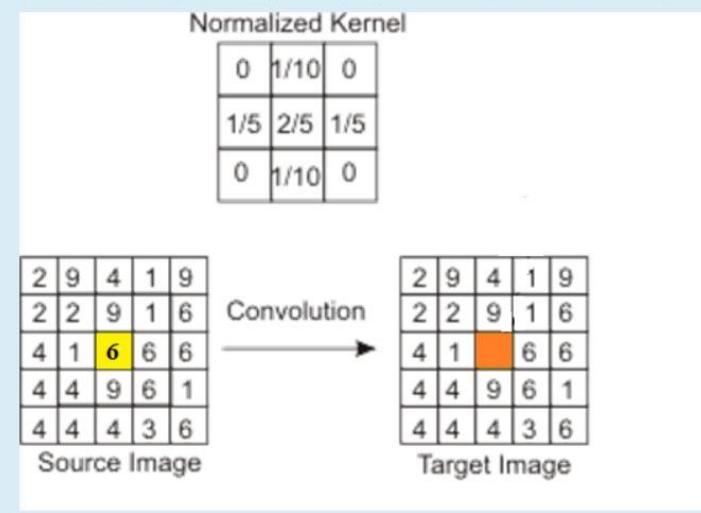
Care este valoarea lipsă din imaginea target aplicând kernelul normalizat centrat pe elementul de culoare galbenă din imaginea sursă?



Alegeți o opțiune:

- a. 4.4
- b. 2/5
- c. 3.4
- d. 4

Care este valoarea lipsă din imaginea target aplicând kernelul normalizat centrat pe elementul de culoare galbenă din imaginea sursă?



Alegeți o opțiune:

- a. 3.4
- b. 4.4
- c. 4/5
- d. 5.6

TEORIE

1. Sunetul - formate de stocare

- formate fara compresie: WAV, AIFF
- formate de compresie fara pierderi: APE, MPEG4, FLAC
- formate de compresie cu pierderi: MP3, AAC, WMA, OGG

Astept continuare...

2. Tehnici de animatie

- Tehnica filmului
- Tehnica cadrelor cheie
- Tehnica schimbarii de culoare
- Tehnica schimbarii de forma

Astept continuare...

3. Grafica raster (BITMAP) - reprezentare si caracteristici

Reprezentata ca o matrice de puncte

Fiecare punct(pixel) retine informatii despre culoarea sa

Culorile sunt stocate conform modelului de culoare al imaginii

Caracteristici:

1. Rezolutia : reprezinta nr de linii si coloane ale matricii de reprezentare sau numarul total de pixeli
2. Adancimea de culoare : cat de multa informatie este retinuta despre culoare

Utilizare:

- Reprezentarea imaginilor pe monitor
- Reprezentarea capturilor externe

Avantaje:

- Poate reprezenta orice fel de imagine

Dezavantaje:

- Nu este scalabila
- Informatia este saracocioasa, nu tine cont de semantica
- Dimensiune mare

Formate: BMP, JPEG, GIF, TIFF

TEHNICI DE ANIMATIE | GRAFICA RASTER (BITMAP)

SUNET – FORMATE DE STOCARE – COMPRESIE / FARÀ COMPRESIE

GRAFICA RASTER (BITMAP) - Reprezentata ca o matrice de puncte

Fiecare punct (pixel) retine informatii despre culoarea sa

Culorile sunt stocate conform modelului de culoare al imaginii

Caracteristici:

1. Rezolutia : reprezinta nr de linii si coloane ale matricii de reprezentare sau numarul total de pixeli
2. Adancimea de culoare : cat de multa informatie este retinuta despre culoare

Utilizare:

-Reprezentarea imaginilor pe monitor

-Reprezentarea capturilor externe

A/D

- + poate reprezenta orice fel de imagine
- nu este scalabila
- informatia este saracicioasa, nu tine cont de semantica
- dimensiune mare

Formate: BMP / JPEG / GIF / TIFF

SUNETUL

TEHNICI DE ANIMATIE

MODELE DE CULOARE

Model de culoare

Model matematic care descrie modalitatea de reprezentare a culorilor sub formă de tupluri numerice

Exemple: RGB, CMY

Spațiu de culori

Modelul de culoare împreună cu instrucțiunile de reprezentare fizică, Exemple: sRGB, AdobeRGB, Pantone

Caracteristici

Țin seama de modalitatea de percepere a luminii de către ochiul uman, Culori de bază albastru (S), verde (M) și roșu (L)

MODELUL ADITIV

Bazat pe culorile de bază roșu, verde și albastru, Culoarea variază în funcție de dispozitiv, (în lipsa unui spațiu de culoare)

Spatii de culoare RGB

sRGB - Dezvoltat de HP și Microsoft, Standard pentru monitoare / imprimante / web, Utilizat ca standard implicit

AdobeRGB - Dezvoltat de Adobe, Acoperă aproape complet spațiul de culori CMYK

MODELUL HSL

Reprezentare sub formă de coordonate cilindrice, Bazat pe aceleași culori de bază: Red – 0grade, Green – 120grade, Blue – 240grade, Axa centrală – tonuri de gri, Utilizat în special pentru selecție de culoare

Modelul substractiv

Modelul CMY(K) - Culori utilizate: cyan, magenta, yellow, Maschează culorile pe o suprafață albă, Utilizare: materiale tipărite

CSS – SPECIFICATOR CULORI

Format hexazecimal: #rgb sau #rrggbb , r,g,b sunt cifre în baza 16

Format RGB: `rgb(red, green, blue)` sau `rgba(red, green, blue, alpha)`, red, green, blue – numere de la 0 la 255 sau procente, alpha – număr între 0 – transparent și 1 – opac sau procent

Format HSL: `hsl(hue, saturation, lightness)` sau `hsla(hue, saturation, lightness, alpha)`, hue – număr de la 0 la 360, saturation, lightness – procente, alpha – număr între 0 – transparent și 1 – opac sau procent

PALETE DE CULORI

Tabel de corespondență: index – tuplu (R, G, B), Utilizare: Reducerea cantității de informație necesară pentru reprezentarea culorilor, Pretabilă pentru imagini cu un număr redus de culori (exemplu: diagrame fără gradienți)

GRAFICA RASTER

Grafica raster (bitmap, matriceală): Reprezentare sub formă de matrice de puncte, Fiecare punct (denumit pixel) stochează informația de culoare, Culorile sunt stocate conform unui model de culoare, direct sau prin intermediul unei palete de culori

Caracteristici: Rezoluție (numărul de linii și coloane stocate în matrice, numărul total de pixeli sau densitate), Adâncime de culoare (cantitatea de informație stocată de către fiecare pixel)

Utilizare: Reprezentare imagine pe monitor, Captare imagini din surse externe

Avantaje și dezavantaje: Poate reprezenta orice imagine, Codaj sărac în informație (nu ia în considerare semantica imaginii), Dimensiune mare, Nu se pot adapta unei scări variabile de vizualizare

FORMATE DE STOCARE

BMP (Microsoft Windows Bitmap): Formatul standard de stocare pe platforma Microsoft Windows, Suportă date necomprimate sau comprimate folosind algoritmul RLE, Monocromă sau în culori pe 4, 8, 16, 24 sau 32 de biți, Suportă palete de culori

JPEG (Joint Photographic Experts Group): Stocare comprimată cu pierdere de informație conform standardului JPEG, Rate de compresie diferite selectabile de către utilizator, Utilizat pentru imagini fotografice (cu gradații fine de culoare),

Nu este potrivit pentru text, linii sau alte imagini care prezintă un contrast foarte mare, Editări multiple (se pierde calitate la fiecare etapă de compresie / decompresie)

GIF (Graphics Interchange Format): Folosit în special pentru transferul imaginilor de maxim 64K x 64K, Pretabil pentru diagrame, text logo-uri (contrast puternic și număr limitat de culori), Suportă maxim 256 culori prin intermediul unei palete de culori, Poate stoca mai multe cadre (pentru animație), Algoritm de compresie fără pierdere de informație Lempel-Ziv-Welch (LZW)

TIFF (Tag Image File Format): Format portabil și extensibil utilizat în special pentru imagini scanate, Suportă stocarea mai multor imagini într-un singur fișier

Suportă mai mulți algoritmi de compresie (RLE, LZW sau JPEG)

Compresia RLE: (nr aparitii, valoare), Rată mică de compresie, Se pretează pentru imagini cu zone mari de aceeași culoare, Utilizat în special pentru fișiere BMP cu paletă de culori

Compresia LRW: Algoritm de compresie universal bazat pe dicționar,

Descriere compresie: Se construiește dicționarul inițial (toate șirurile de lungime 1), Se caută cel mai lung șir W din dicționar care se potrivește cu șirul de la intrare

Se elimină W din șirul de intrare, Se adaugă W urmat de următorul caracter în dicționar, Se continuă cu pasul 2

Decompresie: se parcurge șirul codificat și se reconstruiește dinamic dicționarul

Utilizat pentru fișiere de tip GIF

Compresia Huffman: Algoritm universal de compresie, Codificare optimă de lungime variabilă pentru fiecare simbol în funcție de frecvența de apariție, Datele salvate: dicționarul, datele originale recodificate; Decodificare: translație simbol cu simbol pe baza dicționarului salvat.

Compresia JPEG: Compresie specializată cu pierdere de informație, Rezultate foarte bune pentru fotografii (variații fine de luminozitate și culoare)

Tipuri de compresie: **secvențial** – codaj bazat pe transformarea cosinus discretă cu blocurile procesate în ordinea apariției; **progresiv** – codaj bazat pe transformarea cosinus discretă cu blocurile procesate prin mai multe treceri asupra imaginii; **progresiv fără pierdere** – folosește doar algoritmi de compresie fără pierdere de informație, **progresiv ierarhic** – codifică imaginea la rezoluții din ce în ce mai mari

Etape: 1. Translatarea modului de culoare din RGB în Y'CBCR, 2. Reducerea rezoluției pentru componentele CB și CR, 3. Imaginea se descompune în blocuri de dimensiune 8x8 pixeli, 4. Se aplică transformata cosinus discretă pe fiecare bloc în parte, 5. Aplicarea matricei de cuantizare (pierdere de informație), 6. Blocurile rezultate în urma cuantizării sunt comprimate folosind RLE și Huffman;
Decodificare: se aplică pașii în ordine inversă

GRAFICA VECTORIALA

Bazată pe descrierea matematică a obiectelor componente ale imaginii

Avantaje: Menține semantica – editare la nivel de obiect graphic, Dimensiune redusă, Independente de scara de vizualizare

Dezavantaj: Nu poate reprezenta fidel orice fel de informație

Formate de stocare: SVG (Scalable Vector Graphics) - Format generic bazat pe XML pentru reprezentări vectoriale 2D, Suportă animație și interactivitate

DXF (Drawing Exchange Format)-formatul vectorial lansat de firma Autodesk pentru produsul software AutoCAD

EPS (Encapsulated Post Script) - formatul firmei Adobe pentru imagini vectoriale, se bazează pe un limbaj de descriere numit Post Script

SHP (Shapefile) - formatul firmei ESRI pentru descrierea datelor spațiale de tip: punct, polilinie și polygon, utilizat la reprezentarea elementelor geografice în sisteme de tip GIS

ANIMATIE

Modificarea rapidă a imaginii vizualizate prin modificarea poziției, formei sau dimensiunii unui obiect din imagine; Stocarea numerică a animației presupune reținerea elementelor independente ce compun mișcarea în raport cu factorul timp.Crearea iluziei de mișcare se realizează prin afișarea rapidă de imagini statice ușor modificate

Tehnici principale:Tehnica filmului, Cadre cheie, Schimbarea culorii

SUNETUL

Reprezentare:Axa X: timp, Axa Y: presiune (0 – presiunea aerului în repaus), Amplitudine: măsoară dimensiunea vibrației / volumul sunetului, Frecvență: măsoară viteza vibrației / tonul sunetului

Numerizarea sunetului:Presupune stocarea și prelucrarea sunetului în format digital; Etape:Convertirea sunetului în semnal electric, Eșantionarea și quanticarea semnalului, Stocarea informației numerice pe un suport de memorie externă conform unui format

Avantaje: Stocare mai ușoară, Permite analiza și procesarea numerică a sunetului, nu se degradează în timp sau la copieri repetate

Eșantionare: Prin eșantionare se înțelege procesul de segmentare, cu o perioadă fixă, a semnalului audio analog.Frecvența de eșantionare – rezoluția orizontală. Se determină pe baza teoremei lui Nyquist (minim dublul frecvenței maxime a sunetului)Rate de eșantionare uzuale: 8 kHz – semnal telefonic, 11 kHz – radio AM, 22 kHz – radio FM, 44 kHz – audio CD

Cuantificare: Pp asocierea unei valori numerice corespunzătoare amplitudinii semnalului pentru fiecare interval de timp.Calitatea este influențată de numărul de biți alocați pentru fiecare eșantion (uzual 8 sau 16 biți pentru stocare și 16 – 32 pentru procesare). Redarea sunetului digital:Se reconstruiește sinusoida originală prin interpolarea valorilor numerice stocate, Prin intermediul unui convertor digital

Formate audio:**WAVE** – formatul standard de fișier audio pentru Microsoft și IBM; conține sunet în reprezentare PCM necomprimat;**AIFF** (Audio Interchange File Format) – formatul standard pentru audio digital utilizat pe platformele Apple (variante: necomprimat / comprimat);**MPEG** (Moving Picture Experts

Group) Audio - format standard pentru sunetul digital comprimat; parte a standardului MPEG de codificare a semnalului audio-video; cea mai cunoscută variantă a lui este MP3.

Compresia:Cel mai utilizat algoritm de compresie: MPEG-1 sau 2 Audio Layer III (MP3), Folosește codificare perceptuală - Elimină din rezultat sunetele care nu pot fi percepute de către urechea umană, Sunetele imperceptibile sunt eliminate pe baza unui mode psihoacoustic care exploatează fenomenele de:Mascare a frecvențelor, Mascare temporală

Mascarea frecvențelor: Sunt eliminate sunetele cu frecvență mai mare de 16-18 KHz, Sunt eliminate sunetele de intensitate scazută, care apar concomitent cu sunete de intensitate înaltă, dacă sunt în benzi de frecvență alăturate (cele cu intensitate scazută sunt măcate de cele cu intensitate înaltă)

Mascarea temporală: Se elimină sunetele de intensitate mică care urmează după sunete de intensitate puternică în cadrul unui interval de timp, Sunetele de intensitate mică nu pot fi percepute după sunete de intensitate puternica datorită inerției timpanului

Compresia MP3- etape: 1. Utilizarea de filtre pentru separarea sunetului în 32 sub-benzi de frecvență, 2. Aplicarea modelului psiho-acustic pentru eliminarea sunetelor imperceptibile, 3. Determinarea numărului de biți pentru coeficienți, 4. Prelucrarea valorilor obținute și compunerea fluxului final de biți

VIDEO

Video digital – cuprinde totalitatea tehniciilor de captură, procesare și stocare a imaginilor în mișcare (precum și a sunetului asociat) prin intermediul unui dispozitiv de calcul.

Avantaje video digital:Poate fi procesat prin intermediul calculatorului, Păstrare în timp și rezistență la copieri repetitive, Poate fi transmis la distanță

Caracteristici:Rezoluția, Spațiul de culoare și numărul de biți per pixel, Numărul de cadre pe secundă, Modul de afișare (întrețesut sau progresiv), Calitatea compresiei

Formate: Container – specifică structura de stocare a componentelor video (imagine + audio) și a datelor asociate (metadate, subtitrări, ...)

Advanced Systems Format – ASF: container dezvoltat de Microsoft care poate conține fluxuri codate cu orice codec (Extensii: .ASF, .WMA, .WMV), **Audio Video Interleave – AVI:** container mai vechi dezvoltat de Microsoft pe baza Resource **Interchange File Format – RIFF** (stochează datele în secțiuni identificate prin markere FourCC), **MP4 – MPEG-4 Part 14:** dezvoltat de către Motion Pictures Expert Group și utilizat inițial de către QuickTime (video H.264, audio AAC),

AVCHD – format utilizat în special de către camerele video (video H.264 AVC și sunet AC3 sau PCM), **Matroska / OGG:** formate deschise; pot conține mai multe fluxuri audio / video

Codec – specifică modalitatea de compresie / decompresie pentru un flux video / audio în cadrul unui container

H.264 / MPEG-4 AVC – cel mai popular (utilizat pentru Web, BluRay, camere video), **H.262 / MPEG-2** – formatul standard pentru DVD, **Windows Media Video** – format dezvoltat de către Microsoft, **MJPEG (Motion JPEG)** – format mai vechi bazat pe compresia JPEG

Compresia: Se bazează pe reducerea redundanței din cadrul fluxului video

Redundanță spațială (intra-cadru) - tipul de redundanță identificat și eliminat de algoritmii de compresie a imaginilor, Redundanță temporală (inter-cadru) - redundanță identificată între două cadre consecutive (de exemplu, prin compararea a două cadre se observă că majoritatea pixelilor își păstrează valoarea)

Algoritm de compresie video

Hibrid - Transformata Cosinus Discretă – similar JPEG pentru reducerea redundanței spațiale, Codaj Huffman – pentru comprimarea coeficienților TCD, Codificarea mișcării – pentru reducerea redundanței temporale, Codaj **RLEAsimetric** - Timpul de codare este mult mai mare decât cel de decodare

Etape compresie: Împărțirea imaginii în blocuri: 16x16 luminanță, 8x8 crominanță (culoare); Compresie pe baza DCT pentru reducere spațială, Aplicarea tehniciilor de compensare a mișcării pentru reducere temporală, Faza finală de codare pe două dimensiuni folosind Run Length Encoding

Tipuri de cadre: <I> Intra-picture/frame/image - Cadrele cheie, Necesare pentru căutare și poziționare, Compresie moderată

<P> Predicted pictures - Codate cu referință la un cadru anterior, Folosite ca referință pentru cadre ulterioare

 Bi-directional prediction (interpolated pictures) - Necesită cadre anterioare și viitoare pentru refacere, Compresie mare

NOTITE SEMINAR MULTIMEDIA

SEMINAR 2:

1. Adaugare fisier css extern:

```
<link rel="stylesheet" type="text/css" href="agenda.css">
```

2. Adaugare fisier javascript:

```
<script type="text/javascript"> ->daca scriut direct aici  
<script src="cod.js" type="text/javascript"></script> ->daca iau din alt fisier
```

3. Tabel:

```
<table>  
  <caption>Person List </caption>  
  <thead> //header  
    <tr>  
      <th>Last Name</th> //celula din header  
      <th>First Name</th>  
      <th>Phone</th>  
    </tr>  
  </thead>  
  <tbody> //body  
    <tr>  
      <td>Popescu</td> //celula din body  
      <td>Ion</td>  
      <td>0732555</td>  
    </tr>  
  </tbody>  
  <tfoot> //footer  
    <tr>  
      <td colspan="3">Number of persons:1 </td> //ocupa 3 coloane  
    </tr>  
  </tfoot>  
</table>
```

4. Formular:

```
<form action="#">
<label for="lastName">Last name:</label>
<input type="text" id="lastName" name="lastName" placeholder="last name">
<label for="phone">Phone:</label>
<input type="tel" id="phone" name="phone" placeholder="phone">

<input type="button" value="Add person">
</form>
```

5. CSS:

Selector de tip element:

Body, h1, table, p, etc...

Color->culoarea textului

Background-color->culoarea fundalului

Font-family:tahoma->fontul

Font-size:11pt; ->marime font

Font-weight:bold ->stil font

Text-align:left ->orientare in pagina

Border: 1px solid black

Width: 60%

Border-radius:5ps; ->colturi rotunjite

input[type="button"] ->selector

SEMINAR 3:

1. Validare de null:

```
if (lastName.value === "" || lastName.value == null)
```

2. Validare doar cu cifre:

```
if (!/[0-9]+$/ .test(phone.value))
```

3. Adaugare linii in tabel folosind javascript:

```
//adaugam randuri noi in tabel cu informatia citita
```

```
let tr = document.createElement("tr"); //I. fac randul dorit
```

```

let tdLastName = document.createElement("td"); //1. se face coloana
tdLastName.innerText = lastName; //2. Se pune text in coloana noua
tr.appendChild(tdLastName); //3. Se lipeste coloana la randul creeat

let tdDelete = document.createElement("td")
let Button = document.createElement("input")
Button.type="button"
Button.value="Delete"

Button.addEventListener("click",deletePerson2);
tdDelete.appendChild(Button)

tr.appendChild(tdDelete);

let tBody = document.querySelector("tbody"); // II. selectam body ptr a lipii la el nou
rand creeat
tBody.appendChild(tr); // III. Punem randul in body

```

4. Sterge un rand din tabel:

```

function deletePerson2(){
    let input=this;
    let tdToDelete=input.parentNode;
    let trToDelete=tdToDelete.parentNode;
    let tBody = document.querySelector("tbody");
    tBody.removeChild(trToDelete);
    document.getElementById("noOfPersons").innerText
    =document.getElementById("myTable").rows.length-2;
}

```

SEMINAR 4: CANVAS-BAR CHART

1. Canvas-lucru initial:

```

//inainte de a lucra cu un canvas trebuie sa obtinem un context
let context = this.canvas.getContext("2d")

context.lineWidth = 2; //ptr grosimea liniilor

```

```

context.textAlign="center" //ptr aliniera in cadrul canvasului

context.strokeStyle ="#dedede" //ptr a schimba culoarea contur
context.fillStyle = "#dedede" //ptr a schimba culoarea de umplere
context.fillRect(0, 0, this.canvas.width, this.canvas.height) //ptr a umple un dreptunghi, x,y-
coordonatele de la care desenam, w,h- latime si inaltime

```

2. Desenare grafic cu bare dupa un vector de valori:

```

draw(values){
let context = this.canvas.getContext("2d");

let barWidth = this.canvas.width / values.length;

let maxValue = Math.max(...values)
let f = this.canvas.height / maxValue;

for (let i = 0; i < values.length; i++) {

let barHeight = values[i] * f * 0.9;
let barX = i * barWidth;
let barY = this.canvas.height - barHeight;

let barWidthVisible=barWidth * 0.9

context.fillStyle = "#FF0000"
context.fillRect(barX, barY, barWidthVisible, barHeight)
context.strokeRect(barX, barY, barWidthVisible, barHeight)

context.fillStyle="#000000"
context.fillText(values[i],barX+barWidthVisible/2,barY-10)
}

```

3. Download imagine desenata pe canvas la click pe canvas:

HTML:

```

<a href="#" download="barChart">

<canvas id="canvas" style="width: 800px; height: 600px;">
Your browser does not support the canvas element!

```

```
</canvas>  
</a>
```

JS:

```
canvas.addEventListener("click", function(){  
    let a = this.parentNode;// this este elementul care genereaza click deci canvas si canvas  
    este pus intr-un link a asadar this.parentNode este linkul a  
    let dataUrl=this.toDataURL("image/png")  
    console.log(dataUrl);  
    a.href=dataUrl;  
})
```

4. Eveniment de click pe un element:

```
btnDrawChart.addEventListener("click", function () {...});
```

5. Extragere valoare dintr-un camp:

```
let tbValues = document.getElementById('tbValue');  
let values = tbValues.value;
```

6. Transformare String de forma „[12,2,24]” intr-un vector cu elementele 12,2,24:

```
let valuesArray = eval('[' + values + ']');  
barChart.draw(valuesArray)
```

SEMINAR 5 – CANVAS – HISTOGRAMA DUPA POZA

1. Desenare histograma dupa un set de valori date in vector:

```
draw(values){  
    let context = this.canvas.getContext("2d");  
  
    let maxValue = Math.max(...values)  
    let f = this.canvas.height / maxValue;  
  
    let barWidth = this.canvas.width / values.length;  
  
    context.save();  
    context.rotate(Math.PI);//rotatie la 180 de grade
```

```

context.translate(0,-this.canvas.height)//translate pe verticala, in sus, cu valoarea inaltimei

context.scale(-1,f); // -1 adica face flipp pe verticala, oglinda

for(let i=0;i<values.length;i++)
{
    context.fillRect(i*barWidth,0,barWidth*0.9,values[i]);
}
context.clearRect(0,0,this.canvas.width,this.canvas.height)
context.restore();
}

```

2. `scale(-1, 1)` to flip the context horizontally and `scale(1, -1)` to flip it vertically. The `translate()` method adds a translation transformation to the current matrix by moving the canvas and its origin `x` units horizontally and `y` units vertically on the grid.

3. Drag and drop imagine pe canvas:

```

document.addEventListener("dragover", function (e) {
    e.preventDefault();
})
document.addEventListener("drop", function (e) {

    e.preventDefault();

    let data = e.dataTransfer;
    let files = data.files;

    let fileReader = new FileReader();
fileReader.addEventListener("load", function (e) {

    let dataUrl = e.target.result;// e.target <=> fileReader //refera obiectul de tip
filereader
    let img = document.createElement("img");
img.addEventListener("load", function (e) {

        canvasImage.width = img.naturalWidth;
        canvasImage.height = img.naturalHeight;

        let context = canvasImage.getContext("2d");
        context.drawImage(img, 0, 0);
    })
})
})

```

```

        })
        img.src = dataUrl;

    });
    fileReader.readAsDataURL(files[0]);
}
})

```

4. Extragere pixeli din imaginea de pe canvas – ptr histograma

```

let imageData = context.getImageData(0, 0, canvasImage.width, canvasImage.height)
let data = imageData.data;

for (let i = 0; i < data.length; i += 4) { // fiecare pixel are 4 pozitii ptr canalele rgb
    let r = data[i]; // rosu
    let g = data[i + 1]; // verde
    let b = data[i + 2]; // albastru
    let a = data[i + 3]; // transparenta, daca exista

    let average = Math.round((r + g + b) / 3);

    v[average]++;
}

}

```

SEMINAR 6 – CANVAS – FILTRE PE IMAGINE

1. Tab de „Adauga fisier” care deschide file explorer ptr incarcare imagine:

HTML:

```
<input id="fileBrowser" type="file" accept="image/*">
```

JS:

```
document.getElementById("fileBrowser").addEventListener("change", function (ev) {
    const files = ev.target.files;

    const reader = new FileReader();
    reader.addEventListener("load", function (ev) {
```

```

const dataURL = ev.target.result;

const img = document.createElement("img");
img.addEventListener("load", function (ev) {

    app.visibleCanvas.width = img.naturalWidth;
    app.visibleCanvas.height = img.naturalHeight;
const oContext = app.offscreenCanvas.getContext("2d");
    oContext.drawImage(ev.target, 0, 0);
})
img.src = dataURL;
})
reader.readAsDataURL(files[0]);
});

```

2. Filtre:

- *Greyscale*:

```

for (let i = 0; i < data.length; i += 4) {
    const r = data[i];
    const g = data[i + 1];
    const b = data[i + 2];
    // const a=data[i+3];

    const avg = Math.round((r + g + b) / 3);
    data[i] = data[i + 1] = data[i + 2] = avg;
}

```

- *Threshold*

```

for (let i = 0; i < data.length; i += 4) {
const r = data[i];
    const g = data[i + 1];
    const b = data[i + 2];
    // const a=data[i+3];

    const v = (0.2126 * r + 0.7152 * g + 0.0722 * b >= threshold) ? 255 : 0;
    data[i] = data[i + 1] = data[i + 2] = v;
}

```

- *Sepia*

```

for (let i = 0; i < data.length; i += 4) {
    const r = data[i];
    const g = data[i + 1];
    const b = data[i + 2];
    data[i] = (r * .393) + (g * .769) + (b * .189);
    data[i + 1] = (r * .349) + (g * .686) + (b * .168)
    data[i + 2] = (r * .272) + (g * .534) + (b * .131)
}
• Invert:
for (let i = 0; i < data.length; i += 4) {
    const r = data[i];
    const g = data[i + 1];
    const b = data[i + 2];
    data[i] = 255 - r;
    data[i + 1] = 255 - g;
    data[i + 2] = 255 - b;
}
• Pixelate:
const blocksize = 10;
const oContext = app.offscreenCanvas.getContext("2d");
const vContext = app.visibleCanvas.getContext("2d");

for (var x = 1; x < app.offscreenCanvas.width; x += blocksize) {
    for (var y = 1; y < app.offscreenCanvas.height; y += blocksize) {
        var pixel = oContext.getImageData(x, y, 1, 1);
        vContext.fillStyle = "rgb(" + pixel.data[0] + "," + pixel.data[1] + "," + pixel.data[2] +
    ")";
        vContext.fillRect(x, y, x + blocksize - 1, y + blocksize - 1);
    }
}
• 2Channels:
for (let i = 0; i < data.length; i += 4) {

    const g = data[i + 1];
    data[i + 2] = g;
}
• Red:
for (let i = 0; i < data.length; i += 4) {

    data[i + 1] = 0;
    data[i + 2] = 0;
}

```

3. Cum sa selectezi mai multe butoane care au aceeasi proprietate:

HTML:

```
<button data-effect="normal">Normal</button>
<button data-effect="grayscale">Grayscale</button>
```

JS:

```
document.querySelectorAll("button[data-effect]");
```

SEMINAR 7 – CANVAS – FILTRE PE IMAGINE 2

1. Sa se faca butoanele cu scris rosu cand sunt apasate:

CSS:

```
.active{
    color:red;
}
```

JS:

```
const previousButton = document.querySelector(".active");
if (previousButton != null) {
    previousButton.classList.remove("active");
}
button.classList.add("active");
```

2. Pozitionare div in mijlocul ecranului, sus, folosind css:

```
<div style="
background-color:rgba(200,200,200,0.8);
padding: 10px; position: fixed; top:10px; left:50%; transform: translate(-50%,0);">
```

3. Facem un buton invizibil cu css:

HTML: <button id="btnDownload" style="display: none;">Download</button>
SAU JS: btn.style.display = "none";

4. Aplicare filtru doar pe jumatarea stanga a pozei-preluare pixeli in sectiuni, nu in totalitate

```
const oContext = offscreenCanvas.getContext("2d");
```

```

const imageData = oContext.getImageData(0, 0, offscreenCanvas.width,
offscreenCanvas.height);
const data = imageData.data;

for (let y = 0; y < offscreenCanvas.height; y++) {
    for (let x = 0; x < offscreenCanvas.width/2; x++) {
        const i = (y * (offscreenCanvas.width) * 4) + x * 4;

        const r = data[i];
        const g = data[i+1];
        const b = data[i+2];
        const transparency = data[i+3];
        const average = Math.round((r + g + b) / 3);

        data[i] = average;
        data[i+1] = average;
        data[i+2] = average;
    }
}

```

5. Filtru Darker si Brighter cu o valoare data:

- *Darker:*

```

for (let i = 0; i < data.length; i += 4) {
    const r = data[i];
    const g = data[i + 1];
    const b = data[i + 2];

```

```

    data[i] = r - v;
    data[i + 1] = g - v;
    data[i + 2] = b - v;
}

```

- *Brighter:*

```

    for (let i = 0; i < data.length; i += 4) {
        const r = data[i];
        const g = data[i + 1];
        const b = data[i + 2];

```

```

        data[i] = r + v;
        data[i + 1] = g + v;
        data[i + 2] = b + v;
    }
}

```

6. Buton pentru download imagine de pe canvas:

```

document.querySelector("#btnDownload").addEventListener("click", (ev) => {
  const a = document.createElement("a");
  a.href = app.visibleCanvas.toDataURL();
  a.download = "output.png";
  a.click();
})

```

7. Preluare culoare din punctul in care ne aflam cu mouse-ul:

```

app.visibleCanvas.addEventListener("mousemove", (ev) => {
  const x = ev.offsetX * app.visibleCanvas.width / app.visibleCanvas.clientWidth;
  const y = ev.offsetY * app.visibleCanvas.height / app.visibleCanvas.clientHeight;

  const vContext = app.visibleCanvas.getContext("2d");
  const imageData = vContext.getImageData(x, y, 1, 1);
  const data = imageData.data;
  const r = data[0];
  const g = data[1];
  const b = data[2];

  const span = document.getElementById("color");
  const color = `rgb(${r}, ${g}, ${b})`;//sau const color='rgb(${r},${g},${b})';

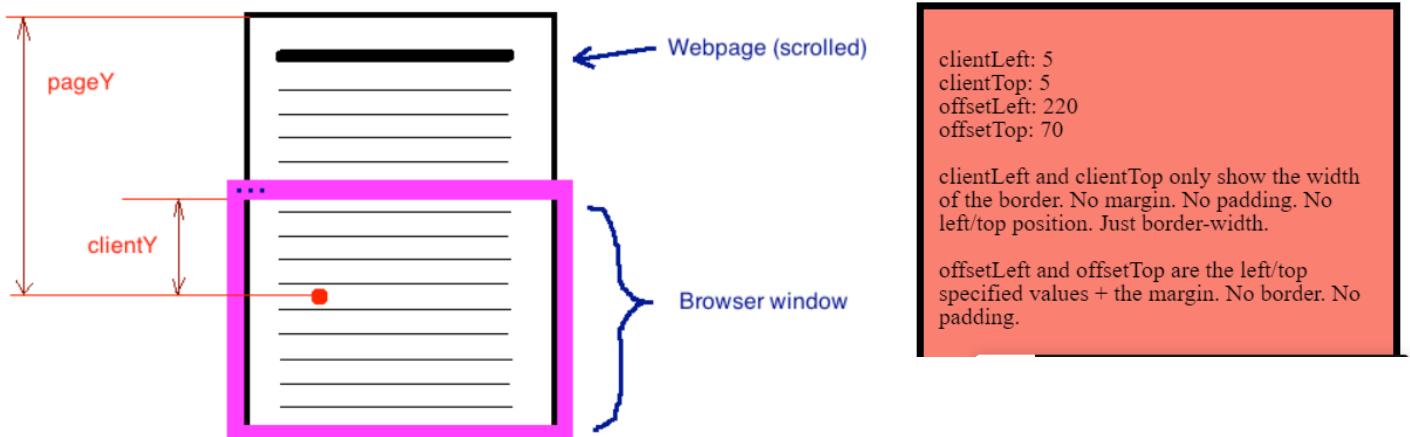
  span.innerText = color;
  span.style.backgroundColor = color;
})

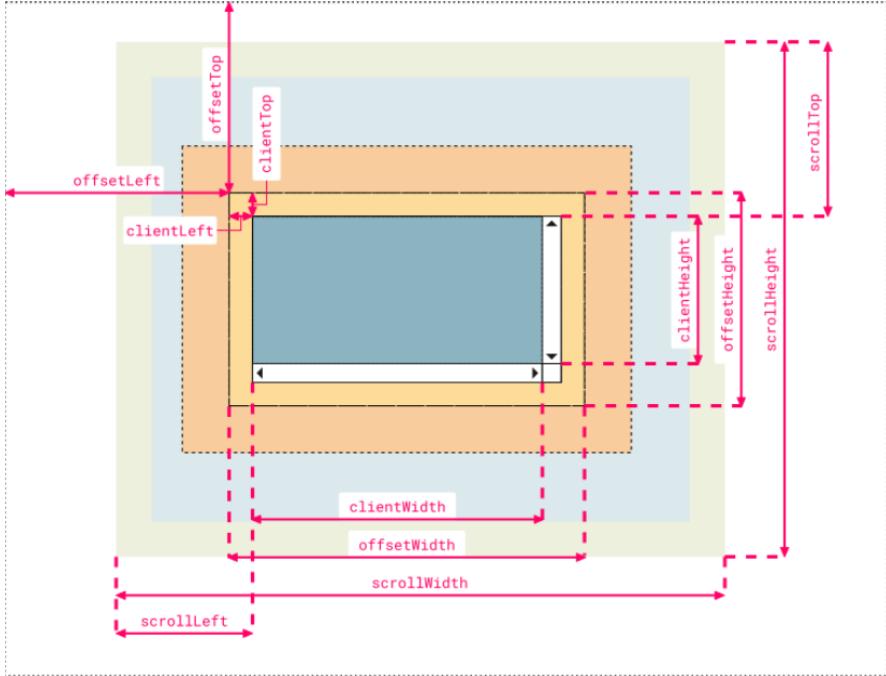
```

8. Pozitie in pagina:

pageX: pozitia in pagina relativa la inceputul intregii pagini

clientX: pozitia in pagina relativa la ce vede utilizatorul(viewport), chiar daca pagina este scrollata si incepe de mai sus.





offsetTop/Left: distanta fata de sus sau stanga relativa la punctul 0.0 din care incepe toata pagina, indiferent daca avem scroll

getBoundingClientRect().top/left: distanta fata de sus sau stanga relativa la fereastra pe care o vede utilizatorul(viewport), chiar daca pagina este scrollata si incepe de mai sus

Daca vrem sa luam pozitia mouse-ului in canvas la evenul de mousedown putem scrie:

```
→pozitie.x = event.pageX -canvas.offsetLeft;
pozitie.y = event.pageY -canvas.offsetTop;
→pozitie.x = event.clientX - canvas.getBoundingClientRect().left;
pozitie.y = event.clientY - canvas.getBoundingClientRect().top;
```

SEMINAR 8 – SVG – BARCHART

1. Schimbam stilul cand suntem deasupra unui element; util pentru butoane:

CSS:

```
.button:hover{  
    fill:yellow;  
}
```

2. Pentru a lucra cu svg avem nevoie de un namespace:

```
this.svgns = "http://www.w3.org/2000/svg";  
this.svg = document.createElementNS(this.svgns, "svg");
```

...PROCESARE SVG...

```
this.element.appendChild(this.svg); //element este un div in html
```

3. Modificari CSS in JS:

```
→this.svg.setAttribute("style", "border: 1px solid black");  
SAU  
→ this.svg.style.borderColor = "black";  
    this.svg.style.borderWidth = "1px";  
    this.svg.style.borderStyle = "solid";
```

4. Desenare rect in SVG:

```
const rect = document.createElementNS(this.svgns, "rect");  
rect.setAttribute("x", 0);  
rect.setAttribute("y", 0);  
rect.setAttribute("width", this.width);  
rect.setAttribute("height", this.height);  
rect.style.fill = "whitesmoke";  
  
this.svg.appendChild(rect);
```

5. Desenare Barchart pe un SVG dupa un vector cu valori:

```

const data=[

    ['Label 1',1],


    ['Label 2',2],


    ['Label 3',3]

]

....  

const barWidth = this.width / this.data.length;
const maxValue = Math.max(...this.data.map((x) => x[1])); //map pt a obtine doar valorile numerice
const f = this.height / maxValue;

for (let i = 0; i < this.data.length; i++) {
    const bar = document.createElementNS(this.svgns, "rect"); //desenare rect
    const text=document.createElementNS(this.svgns,"text"); //desenare text

    const label = this.data[i][0];
    const value = this.data[i][1];

    const barHeight = value * f * 0.9;

    const barX = i * barWidth;
    const barY = this.height - barHeight; //ca sa nu le afiseze de sus

    bar.classList.add("bar"); //ptr prelucrarea in css cu selectorul .bar

    bar.setAttribute("x", barX + barWidth / 4);
    bar.setAttribute("y", barY);
    bar.setAttribute("width", barWidth / 2);
    bar.setAttribute("height", barHeight);

    bar.addEventListener("click", ()=>{
        alert(value);
    });

    text.setAttribute("x", barX+barWidth/4);
    text.setAttribute("y", barY-10);
    const labelValue=document.createTextNode(label);
    text.appendChild(labelValue);
}

```

```

// bar.style.fill = "red"; -> aici daca vrem sa facem aia cu hover pe galben are
specificitatea prea mare si raman rosii
//daca am pune cu bar.setAttribute("fill","red"); //ar fi functionat ptr ca nu mai are
specificitate mai mare decat hover

bar.style.stroke = "black";
//bar.setAttribute("stroke-width", "10px");
bar.style.strokeWidth = "2px";
//sau bar.style["stroke-width"]="2px"

this.svg.appendChild(bar);
this.svg.appendChild(text);

}

```

SEMINAR 9 – AUDIO.1

1. Adaugare librarii externe-Bootstrap si fonturi Cloudflare

```

<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->
<link rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
      integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
      crossorigin="anonymous">

<!--Fonturi-->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css">

```

2. Element de tip audio in HTML:

```

<audio id="audio" src="media/Beethoven-MoonlightSonata.mp3">
    Your browser does not support the "audio" element
</audio>
SAU
<audio
    controls //ne face automat butoane de control
    src="/media/cc0-audio/t-rex-roar.mp3">
    Your browser does not support the audio element.
</audio>

```

3. Buton de play/pause pentru muzica din elementul audio:

```

app.btnExit.addEventListener("click", () => {
    if (app.audio.paused)
        app.audio.play();
    else
        app.audio.pause();
});

```

4. Cum sa afisam durata totala a melodiei si pozitia curenta in melodie:

→functie de formatare float in min si sec ptr afisare corecta:

```

function formatTime(seconds) {

    let minutes = Math.floor(seconds / 60);
    let secondsRemaining = Math.floor(seconds - minutes * 60);

    let result = "" + (minutes < 10 ? "0" : "") + minutes + ":" + (secondsRemaining < 10 ? "0" : "") +
    secondsRemaining;

    return result;
}

```

→eventimentele care trebuie tratate:

```

app.audio.addEventListener("durationchange", () => {//cand se schimba durata totala
    app.spDuration.innerHTML = formatTime(app.audio.duration); // audio.duration ne da
    lungimea totala a melodiei
});

```

```

app.audio.addEventListener("timeupdate", () => {//cand se schimba valoarea din currentTime
    app.spCurrentTime.innerHTML = formatTime(app.audio.currentTime); //audio.currentTime
    ne da pozitia curenta in melodie
});

```

SEMINAR 10 – AUDIO.2->playlist

1. Playlist-ul poate fi o simpla lista in care pastram sursa ptr fiecare melodie cu un atribut:

```

<ul id="playlist" class="list-group">
    <li class="list-group-item" data-url="media/Bolero.mp3">
        Ravel - Bolero
    </li>
    <li class="list-group-item" data-url="media/Beethoven-MoonlightSonata.mp3">
        Beethoven - Moonlight Sonata
    </li>
    <li class="list-group-item" data-url="media/CanoninD.mp3">
        Pachelbel - Canon in D
    </li>
</ul>

```

2. Cum selectam o melodie din playlist ptr a da play la click pe ea:

```

→ const elements = document.querySelectorAll("li[data-url]"); //selectam toate randurile din
lista, adica toate melodiile si ptr fiecare am salvat in atributul data-url link-ul sursa

for (let i = 0; i < elements.length; i++) {
    const element = elements[i];
    const url = element.dataset.url;

    element.addEventListener("click", function () {
        app.play(url);
    })
}

```

```

→app.play = function (url) { //asta e functia de play din aplicatia scrisa de noi

    const previouslySelectedElement = document.querySelector("li.active");
    if (previouslySelectedElement !== null)
        previouslySelectedElement.classList.remove("active");

    const selector = 'li[data-url="' + url + '"]';
    const selectedElement = document.querySelector(selector);

    selectedElement.classList.add("active");

    app.audio.src = url;
    app.audio.play();
}

→app.audio.addEventListener("play", () => { //asta e functia de play a elementului audio
    const selectedElement = document.querySelector("li[data-
url='"+app.audio.getAttribute('src')+']."'");

    selectedElement.classList.add("active");
});

```

3. Buton de +10 secunde in melodie:

```

app.btnForward.addEventListener("click", () => {
    app.audio.currentTime += 10;
});

```

SEMINAR 11 – AUDIO.3 -> MICROFON SI SOUND BAR ANALYZAR-Frequency Bars

1. Dropdown de tip select in HTML:

HTML:

```

<label for="visualisation"> Visualizer settings</label>
<select id="visualisation" >
<option disabled> Choose</option>
<option value="sinewave">Sinewave</option>
<option value="bars">Frequency Bars</option>

```

```

</select>
JS:
display(visualisation) {
    this.audioContext.resume();

    if (visualisation === "bars") {
        this.displayBars();
    } else if (visualisation === "sinewave") {
        this.drawSineWave();
    }
}

```

2. The `MediaDevices.getUserMedia()` method prompts the user for permission to use a media input which produces a `MediaStream` with tracks containing the requested types of media. That stream can include, for example, a video or an audio.

```
stream = await navigator.mediaDevices.getUserMedia;
```

3. Cum sa preluam audio-ul din microfon:

```

→navigator.mediaDevices.getUserMedia({
    audio:true //daca exista un input de tip audio
}).then(function(stream){
    soundAnalyzer.setStreamSource(stream) //atunci stream-ul creat de input se trimit
    mai departe in aceasta metoda care construieste un nod sursa?
})
.catch(function(err){
    alert(err.message);
})

```

→Undeva in constructor :

```

this.audioContext = new AudioContext();
this.analyzerNode = this.audioContext.createAnalyser();
```

```

→setStreamSource(stream){
    const sourceNode = this.audioContext.createMediaStreamSource(stream);
    sourceNode.connect(this.analyserNode);
}
```

!!! daca vrem, putem transmite aici un element audio care are o melodie in el, nu neaparat un stream de la microfon DAR PUNEM MediaElementSource

```
setMediaElementSource(mediaElement) {  
    this.sourceNode = this.audioContext.createMediaElementSource(mediaElement);  
    this.sourceNode.connect(this.analyzerNode);  
}  
https://developer.mozilla.org/en-US/docs/Web/API/Web\_Audio\_API
```

AudioContext

The `AudioContext` interface represents an audio-processing graph built from audio modules linked together, each represented by an `AudioNode`. You need to create an `AudioContext` before you do anything else, as everything happens inside a context.

AnalyserNode

The `AnalyserNode` interface represents a node able to provide real-time frequency and time-domain analysis information, for the purposes of data analysis and visualization.->`audioContext.createAnalyser()`

MediaStreamAudioSourceNode

The `MediaStreamAudioSourceNode` interface represents an audio source consisting of a `MediaStream` (such as a webcam, microphone, or a stream being sent from a remote computer). It is an `AudioNode` that acts as an audio source.->`audioContext.createMediaStreamSource(stream)`

4. Construirea graficului cu bare ptr analiza audio: sound bar analyser-Frequency Bars

```
drawBars(){  
    this.context.fillStyle = "black";  
    this.context.fillRect(0, 0, this.canvas.width, this.canvas.height);  
  
    this.analyserNode.fftSize = 256;  
  
    const bufferLength = this.analyserNode.frequencyBinCount;  
    const buffer = new Uint8Array(bufferLength);  
    this.analyserNode.getByteFrequencyData(buffer);  
  
    const barWidth = this.canvas.width / buffer.length;  
    const f = this.canvas.height / 255;  
  
    this.context.fillStyle = "red";  
    for(let i=0; i< buffer.length; i++){  
        const barHeight = buffer[i] * f;  
        const barX = i*barWidth;  
        const barY = this.canvas.height - barHeight;
```

```

        this.context.fillRect(barX, barY, barWidth, barHeight);
    }

    //setInterval
    requestAnimationFrame(()=> this.drawBars());
}

```

The `window.requestAnimationFrame()` method tells the browser that you wish to perform an animation and requests that the browser calls a specified function to update an animation before the next repaint. The method takes a callback as an argument to be invoked before the repaint.

SEMINAR 12.1 – AUDIO.4 -> MICROFON SI SOUND WAVE ANALYZAR- Sinewave

1. Construirea graficului cu wave ptr analiza audio: sound waveanalyser-Sinewave

```

drawSineWave() {
    let bufferLength = this.analyserNode.fftSize;
    let dataArray = new Float32Array(bufferLength);
    this.analyserNode.getFloatTimeDomainData(dataArray);

    this.context.fillStyle = 'rgb(200, 200, 200)';
    this.context.fillRect(0, 0, this.canvas.width, this.canvas.height);
    this.context.lineWidth = 2;
    this.context.strokeStyle = 'rgb(0, 0, 0)';

    this.context.beginPath();

    let sliceWidth = this.canvas.width * 1.0 / bufferLength;
    let x = 0;
    for (let i = 0; i < bufferLength; i++) {

```

```

let v = dataArray[i] * 200.0;
let y = this.canvas.height / 2 + v;

if (i === 0) {
    this.context.moveTo(x, y);
} else {
    this.context.lineTo(x, y);
}
x += sliceWidth;
}
this.context.lineTo(this.canvas.width, this.canvas.height / 2);
this.context.stroke();

this.drawVisual = requestAnimationFrame(() => this.drawSineWave());
}

```

SEMINAR 12.2 – VIDEO

1. Element de tip video:

```

<video
id="video"
src="media/tears-of-steel-battle-clip-medium.mp4"
muted
controls>
    Not supported
</video>

```

!!!Daca punem video ca atare, va fi vizualizat din prima. Daca il punem intr-un canvas trebuie sa facem context.drawImage(video,0,0);

2. Captura de ecran din video:

```

btnCapture.addEventListener("click", ()=>{
    canvasCapture.width = video.videoWidth//!aparent nu merge cu
    video.clientWidth;
    canvasCapture.height = video.videoHeight;

    const context = canvasCapture.getContext("2d");
    context.drawImage(video,0,0);
});

```

3. Mut si unmute pe video:

```
btnMuted.addEventListener("click", ()=> {
    if(video.muted === true){
        video.muted = false;
        btnMuted.innerHTML = "Mute";
    }
    else{
        video.muted = true;
        btnMuted.innerHTML = "UnMute";
    }
});
```

4. Pause si unpause pe video:

```
btnPlayPause.addEventListener("click", ()=> {
    if(video.paused === true){
        video.play();
        btnPlayPause.innerHTML = "Pause";
    }
    else{
        video.pause();
        btnPlayPause.innerHTML = "Play";
    }
});
```

5. Canvas cat un video:

```
video.addEventListener("canplay", ()=>{
    canvasVideo.width = video.videoWidth; // aparent nu merge cu video.clientWidth;
    canvasVideo.height = video.videoHeight;
});
```

6. Animatie pe video:filtru si secunde

```
function draw(){
    contextVideo.drawImage(video, 0, 0);

    const imageData = contextVideo.getImageData(0, 0,
                                                canvasVideo.width, canvasVideo.height);
    const data = imageData.data; //merge doar pe microsoft edge, nu si pe chrome
```

```

/*for(let i=0; i<data.length; i+=4){
    const average = (data[i] + data[i+1] + data[i+2])/3;
    data[i] = data[i+1] = data[i+2] = average;
};*/

```

→ aplicare filtru

```

for (let y=0;y<canvasVideo.height;y++)
    for (let x=0;x<canvasVideo.width;x+=2)
    {
        let i= y*canvasVideo.width*4+x*4;

        const average=(data[i]+data[i+1]+data[i+2])/3;
        data[i]=data[i+1]=data[i+2];
    }

```

```
contextVideo.putImageData(imageData, 0, 0);
```

```

contextVideo.fillStyle = "white";
contextVideo.font = "100px sans-serif";
contextVideo.textAlign = "center"
contextVideo.fillText(
video.currentTime,
canvasVideo.width/2, 100)

```

```

requestId = requestAnimationFrame(draw); //ptr a desena la fiecare secunda, nu doar
odata
}

```

7. Daca vrem ca animatia cu filtru pe video si scris sa se opreasca atunci cand punem pauza:

```

let requestId=null;
video.addEventListener("play", ()=>{
requestId = requestAnimationFrame(draw);
})
video.addEventListener("pause", ()=>{
cancelAnimationFrame(requestId);
})

```

8. Activare si dezactivare controls

```

const btnControls = document.getElementById("btnControls");
btnControls.addEventListener("click", () => {

```

```
    if (video.controls === true)
        video.controls = false;

    else
        video.controls = true;
});
```

SEMINAR 13-JOC CU CARAMIZI

...

PROIECTUL CU CANVAS EDITOR FOTO

1. Stilul textului

```
editor.context.fillStyle = 'red';

editor.context.font = "20px Georgia"

editor.context.fillText('Trage o imagine pe canvas', editor.canvas.width / 2 - 90, editor.canvas.height /
2, 200);
```

2. Canvas proportionat, pastrarea proportiilor la redimensionare

```
ratio = editor.image.naturalHeight / editor.image.naturalWidth;
editor.canvas.width = 400;
editor.canvas.height = editor.canvas.width * ratio;
editor.context.drawImage(editor.image, 0, 0, editor.canvas.width,
editor.canvas.height);
```

3. Canvas cat imaginea:

```
canvasImage.width = img.naturalWidth;

canvasImage.height = img.naturalHeight;
```

4. Drag and drop:

```
editor.canvas.addEventListener("dragover", function (event) {
    event.preventDefault();
```

```

});;

editor.canvas.addEventListener("drop", function (event) {
    var fisierePrimite = event.dataTransfer.files;
    var fisier = fisierePrimite[0]; //primul fisier

    var cititor = new FileReader();
    cititor.onload = function (event) {

        editor.image = document.createElement("img");
        editor.image.addEventListener("load", function () {

            editor.context.clearRect(0, 0, editor.canvas.width, editor.canvas.height);

            ratio = editor.image.naturalHeight / editor.image.naturalWidth;
            editor.canvas.width = 400;
            editor.canvas.height = editor.canvas.width * ratio;
            editor.context.drawImage(editor.image, 0, 0, editor.canvas.width,
editor.canvas.height);

            salveazaStareCurenta();

        });

        editor.image.src = event.target.result; //cand se termina functia load se deseneaza poza
pe canvas
    };
    cititor.readAsDataURL(fisier); //cand se termina functia load ptr cititor

    event.preventDefault();
});

```

5. Incarcare poza din fileBrowser:

```

<input id="fileBrowser" type="file" accept="image/*">

//INCARCARE POZA DIN FILE BROWSER
document.getElementById("fileBrowser").addEventListener("change", function (event) {
    var fisierePrimite = event.target.files;

    var cititor = new FileReader();
    cititor.addEventListener("load", function (event) {

```

```

editor.image = document.createElement("img");
editor.image.addEventListener("load", function (event) {

    editor.context.drawImage(editor.image, 0, 0, editor.canvas.width,
editor.canvas.height);

});

editor.image.src = event.target.result;
});
cititor.readAsDataURL(fisierePrimite[0]);

});

```

6. DrawImage a treia implementare cu source image:

```

//si desenez in canvas doar zona selectata
editor.context.drawImage(editor.image,
editor.selectiaMea.x, editor.selectiaMea.y,
editor.selectiaMea.width, editor.selectiaMea.height,
0, 0,
editor.canvas.width, editor.canvas.height);

```

7. Desenare elipsa sau cerc

```

let x = canvas.width / 2 - 25;
let y = canvas.height / 2 - 25;
context.beginPath();
context.arc(x, y, 50, 0, 2 * Math.PI);
context.stroke();

let x1 = canvas.width / 2 - 30;
let y1 = canvas.height / 2 - 50;
context.beginPath();
context.ellipse(x1, y1, 60, 100, 0, 0, 2 * Math.PI);
context.stroke();

```

8. Rotit canvas la 180 de grade

```

context.translate(canvas.width, canvas.height);
context.rotate(Math.PI);
context.rotate(Math.PI); //rotatie la 180 de grade

```

```
context.translate(0,-this.canvas.height) //translate pe verticala, in sus, cu valoarea inaltimei
```

9. Rotire scris in canvas la 90 de grade:

```
context.translate(canvas.width, -canvas.height/2);
context.rotate(Math.PI/2)
```

10. Afisare video pe canvas cu filtru gri

```
let requestId = null;
video.addEventListener("play", function () {
    requestId = requestAnimationFrame(draw);
})
video.addEventListener("pause", function () {
    cancelAnimationFrame(requestId);
})

function draw() {
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    context.drawImage(video, 0, 0);

    let imageData = context.getImageData(0, 0, canvas.width, canvas.height);
    let data = imageData.data;

    for (let i = 0; i < data.length; i += 4) {
        const r = data[i];
        const g = data[i + 1];
        const b = data[i + 2];

        const avg = Math.round((r + g + b) / 3);
        data[i] = data[i + 1] = data[i + 2] = avg;
    }

    context.putImageData(imageData, 0, 0);
    requestId = requestAnimationFrame(draw);
}
```

11. Functie cu parametru:

```
buton.addEventListener("click", function () {
    /**
     * @param {HTMLImageElement} img
     */
```

```

        draw(img)

    })
function draw(img) {

    canvas.width = img.naturalWidth;
    canvas.height = img.naturalHeight;
    context.drawImage(img, 0, 0);

    let imageData = context.getImageData(0, 0, canvas.width, canvas.height);
    let data = imageData.data;

    for (let i = 0; i < data.length; i += 4) {
        let r = data[i];
        let g = data[i + 1];

        data[i] = g;
        data[i + 1] = r;
    }
    context.putImageData(imageData, 0, 0);

    context.strokeStyle="green";
    context.strokeRect(canvas.width/2-200, canvas.height/2-300,
    400,600);

}

```

12. Colțul din stanga jos al canvasului

```

for (let x = canvas.height - 200; x < canvas.height; x++) {
    for (let y = 0; y < 200; y++) {

        let i = x * canvas.width * 4 + y * 4;
        let r = data[i];
        let g = data[i + 1];
        let b = data[i + 2];

        data[i] = data[i + 1] = data[i + 2] = Math.round((r + g + b)
/ 3);
    }
}

```

13. Button pe play pause audio:

```
btn.addEventListener("click", function () {
```

```

        if (audio.paused) {
            audio.play();
            btn.innerHTML = "Pause";
        }
        else {
            audio.pause();
            btn.innerHTML = "Play";
        }
    });

```

14. Afisare video pe canvas:

```

video.addEventListener("play", function () {
    requestAnimationFrame(draw);
});

function draw() {

    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;

    context.drawImage(video, 0, 0);

    context.textAlign = "center";
    context.font="25px Georgia"
    context.fillText(video.src,canvas.width/2,canvas.height/2);

    requestAnimationFrame(draw);
}

```

15. Histograma cu rosu

```

btn.addEventListener("click", function () {
    let value = [];
    for (let i = 0; i < 256; i++)
        value[i] = 0;

    canvas.width = img.naturalWidth;
    canvas.height = img.naturalHeight;
    context.drawImage(img, 0, 0);

    let imageData = context.getImageData(0, 0, canvas.width, canvas.height);
    let data = imageData.data;

```

```
for (let i = 0; i < data.length; i += 4) {
    value[data[i]]++;
}

context.clearRect(0, 0, canvas.width, canvas.height);

context.font = '25px Arial'
context.fillStyle = 'green';
context.fillText('Alexandra', 50, 50)

let max = Math.max(...value);
let f = canvas.height / max;

let barWidth = canvas.width / value.length

context.rotate(Math.PI);
context.translate(0, -canvas.height);
context.scale(-1, f);

for (let i = 0; i < value.length; i++) {
    context.fillRect(i * barWidth, 0, barWidth * 0.9, value[i]);
}

})
```

Multimedia

conf. dr. Cristian Ioniță

Resurse / Contact

Resursele necesare se găsesc:

- <https://online.ase.ro> – pe pagina cursului / seminarului
- <http://ase.softmentor.ro> – secțiunea *Multimedia* → *Teme abordate*

Modalități de contact:

- Email: cristian.ionita@ase.ro / crionita@ie.ase.ro / cristian.ionita@softmentor.ro
- Mesaje pe platforma <https://online.ase.ro>

Obiective

Însușirea elementelor teoretice și practice care să permită dezvoltarea de aplicații multimedia

- Cunoașterea facilităților multimedia ale sistemelor de calcul actuale și a instrumentelor software pentru dezvoltarea de aplicații multimedia
- Cunoașterea noțiunilor teoretice necesare pentru procesarea culorilor, imaginilor raster și vectoriale, sunetului și secvențelor video
- Dezvoltarea de abilități de programare a aplicațiilor multimedia în context web

Tehnologii utilizate:

- Limbajele HTML, CSS și JavaScript
- API-urile puse la dispoziție de browser-ele web moderne

Evaluare

50% – evaluare finală

- Examen oral în ultimele două săptămâni

30% – evaluări pe parcurs

- Lucrare săptămâna 4 – manipulare DOM folosind JavaScript (20% din nota finală)
- Lucrare săptămâna 12 – elemente teoretice (10% din nota finală)

20% – proiect

- Individual, tema de proiect se alocă de către profesor, prezentare în săptămâna 12

Condiții de promovare

- Minim nota 5 la evaluarea finală
- Minim nota 5 la evaluarea pe parcurs
- Minim nota 5 la proiect

Teme Proiecte

1. Program de desenare

- Permite adăugarea interactivă și manipularea de figuri geometrice pe o imagine raster și exportul desenului în format raster sau vectorial

2. Editor de imagini

- Permite încărcarea de imagini și efectuarea interactivă de operații (crop, redimensionări, aplicare efecte, afișarea histogramei de culoare, adăugare de text, ...)

3. Player video

- Permite navigarea și editarea playlist-ului, afișarea de subtitrări, aplicarea de efecte în timp real, ...

4. Editor grafică vectorială (SVG)

- Permite editarea unui desen folosind grafică vectorială și exportul sub formă de SVG sau raster

5. Joc – similar Asteroids

6. Vizualizare date

- Vizualizare grafică interactivă pentru date preluate de pe Eurostat

Structura curs

1. Noțiuni generale

- Conceptul de multimedia și clase de aplicații
- Condiții hardware și software
- Multimedia în context web (HTML / CSS / JavaScript)

2. Imagine și animație

- Culoarea și spații de culoare
- Grafica raster
- Desenarea în grafica raster
- Procesarea imaginilor
- Stocarea și compresia imaginii
- Animație
- Grafica vectorială

3. Sunet

- Noțiuni de bază legate de sunet și procesul de numerizare
- Compresia audio
- Utilizarea sunetului în aplicații web (redare, generare și procesare)

4. Video

- Noțiuni generale și compresia
- Utilizarea secvențelor video și captura video în aplicații web

5. Integrarea elementelor multimedia în aplicații web, desktop sau mobile

I. Noțiuni generale

1. Conceptul de multimedia
2. Clase de aplicații multimedia
3. Condiții hardware / software
4. Multimedia în context WEB

1. Conceptul de multimedia

Multimedia din punct de vedere informatic este:

- o combinație de text, imagine, sunet, animație, video
- accesibilă utilizatorului prin intermediul sistemului de calcul

Elemente care au stat la baza dezvoltării conceptului de multimedia:

- conversia analog digital
- compresia datelor

Conversia analog - digital

Resursele utilizate trebuie convertite în format digital pentru:

- stocare
- procesare
- includere în aplicații

Resursele sunt convertite în format analog la redare

Compresia

- apariția algoritmilor specifici (cu pierdere de informație)
- exemple: JPEG, MPEG Audio / Video
- pierdere de informație controlată
 - exploatează limitările perceptiei umane
- proces asimetric

Tehnologii utilizate

- periferice specializate pentru captură imagine / sunet / video
- medii de stocare de mare capacitate
- tehnologii de comunicare la distanță cu bandă largă
- procesoare specializate pentru compresie / decompresie la nivel hardware

Aplicație multimedia

- componentele sunt accesibile prin intermediul unui sistem de calcul
- datele utilizate sunt în format digital (NU analogic)
- elementele sistemului sunt integrate într-o interfață unitară
- grad ridicat de interacțiune cu utilizatorul

Modalități de dezvoltare

Multimedia **authoring**

- includ componente preprogramate ce permit recunoașterea mai multor formate de resurse multimedia, instrumente pentru generarea animațiilor, pentru implementarea conceptelor de hypertext, hypermedia
- accentul cade pe scenariul de derulare a aplicației și pe sincronizarea resurselor
- bazat pe concepte precum axa timpului / carte / diagrame de flux
- Exemple: Flash, PowerPoint

Multimedia **programming**:

- accentul se pune pe procesarea directă a resurselor prin intermediul unui limbaj de programare și a unor biblioteci specializate

2. Clase de aplicații multimedia

Criterii multiple de clasificare

Cele mai importante criterii

- domeniu de utilizare
- grad de interacțiune
- localizarea componentelor

Clasificare în funcție de domeniu

- **Economie** – vizualizarea datelor
- **Educație** – aplicații de e-learning, enciclopedii
- **Publicitate** – aplicații de prezentare
- **Medicina** – procesarea și vizualizarea interactivă a datelor medicale
- **Industrial** – instrumente de proiectare asistată
- **Entertainment** – jocuri, realitate virtuală
- **Sisteme informaticice geografice GIS** – sisteme pentru vizualizarea datelor geo-spațiale
- **Comunicații** – aplicații de tip videoconferință

Alte criterii

Grad de interacțiune

- aplicații interactive
- prezentări statice sau liniare

Localizarea componentelor

- locale
- la distanță

3. Precondiții hardware / software

Echipamente hardware pentru achiziție / procesare:

- imagine
- sunet
- video

Componente software necesare pentru redarea conținutului multimedia

Hardware specializat

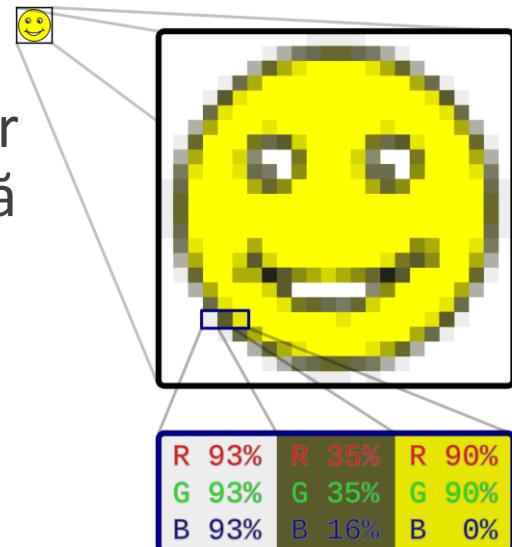
Dispozitive periferice pentru achiziția de **imagini fixe**:

Scanner

- transformă informația luminoasă în informație electrică, iar ulterior aceasta este convertită și salvată sub formă digitală
- tipuri: flatbed / rotativ

Aparat foto digital

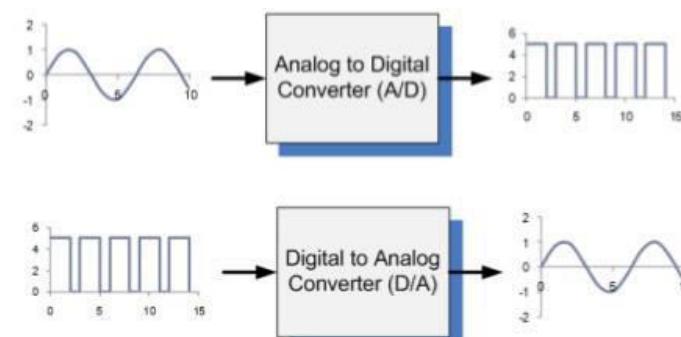
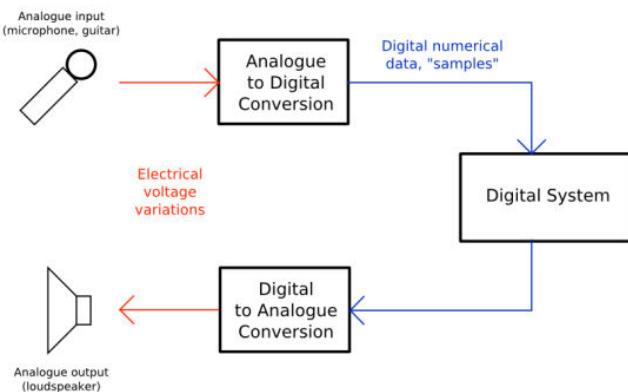
- folosește lentile asemănătoare aparatului foto clasic și un senzor digital pentru transformarea informației luminoase în informație electrică



Hardware specializat

Dispozitive periferice pentru achiziția de sunet:

- placă de sunet: convertor analog – digital și digital – analog



Hardware Specializat

Dispozitive periferice pentru achiziția de **imagini video**:

- placă de achiziție și numerizare video: preluare și numerizare fluxuri audio - video
- camera video digitală: captură directă în format digital (similar aparat foto digital)
- placă TV tuner: preia semnal TV, decodifică și numerizează semnalul

Implementare hardware pentru algoritmi de compresie și compresie în cadrul plăcilor video

Software specializat

Drivere

- Asigură controlul și comunicarea cu perifericele specializate
- Specifice pentru dispozitivul și sistemul de operare
- Permit sistemului de operare să ofere aplicațiilor o interfață uniformă

Extensii ale sistemului de operare

- Biblioteci specializate pentru controlul resurselor multimedia
- Instrumente de bază pentru redarea conținutului multimedia

Software specializat

Produse software specializate pe medii de comunicare

Achiziție și prelucrare de **imagini**

- Permit prelucrarea imaginilor în format raster sau vectorial
- Exemple: Adobe PhotoShop, GIMP, Corel Draw, Adobe Illustrator

Achiziție și prelucrare de **sunet**

- Exemple: Adobe Audition, Audacity, Sony Sound Forge

Achiziție și editare **video**

- Exemple: Adobe Premiere, Blender, Sony Vegas, DaVinci Resolve

Software specializat

Produse software pentru **creație și redare de aplicații multimedia**

Exemple

- Browser web
- Flash
- PowerPoint

4. Multimedia în context WEB

Se dezvoltă odată cu apariția standardului HTML5

Elemente multimedia suportate

- Text
- Imagini
- Animație
- Grafică raster – elementul *canvas*
- Grafică vectorială – SVG
- Sunet – elementul *audio* + *Web Audio API*
- Video – elementul *video*
- Grafică 3D – WebGL

Avantaje

Arhitectură client-server

Client – Web Browser

- Trimitere cererile către server
- Gestionează afişarea conţinutului multimedia şi interacţiunea cu utilizatorul

Server – Web Server

- Preia cererile de la Web Browser
- Trimitere conţinutul (static sau generat dinamic) către browser

Comunicare – protocolul Hypertext Transfer Protocol

- Protocol text
- Mesaje de tip *request* și *response*
- Mesajele sunt trimise prin intermediul protocolului TCP/IP

HTTP Request

Format: comanda (GET / POST / ...)/ parametri / eventual conținut

Cereri inițiate de către browser:

- Introducere adresă de către utilizator
- Navigare către pagină nouă
- Obținere resurse
- Solicitat din cod

Exemplu:

GET /index.html HTTP/1.1

Host: www.test.com

User-Agent:Mozilla/5.0

HTTP Response

Format: status (200 / 404 / 500 /...)/ parametri / conținut

Răspunsul este generat:

- Static (conținutul este preluat din fișier)
- Dinamic (conținutul este generat de către un program)

Exemplu:

HTTP/1.1 200 OK

Content-Type: text/html; charset=UTF-8

Content-Length: 131

<html>

...

</html>

Limbaje utilize

HTML - HyperText Markup Language

- Structura documentului
- Conținutul (de tip text)
- Referințe la alte resurse

CSS - Cascading Style Sheets

- Descrie modalitatea de prezentare a conținutului

JavaScript

- Manipularea dinamică a conținutului și a modalității de prezentare
- Gestionează interacțiunea cu utilizatorul

DOM – Document Object Model

Aplicația este reprezentată intern sub formă de arbore

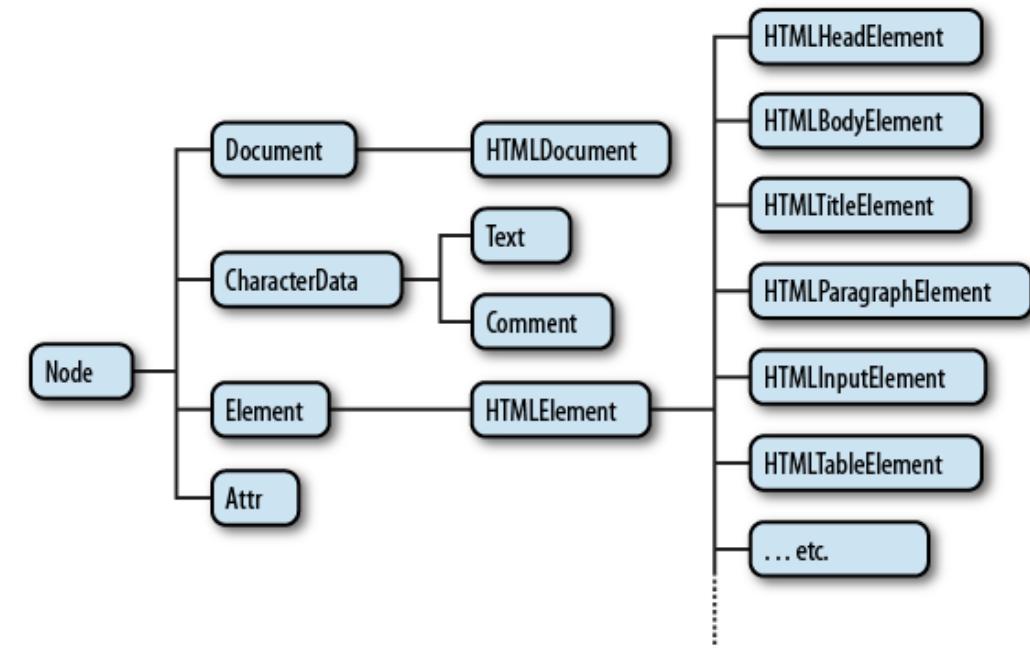
Tipuri de noduri

- Elemente
- Text

Nodurile - create din HTML sau JavaScript

Conțin:

- atribute
- parametri de stil
- funcții JavaScript



Limbajul HTML

Tag

- Instrucțiunea de creare a unui nod
- Format: **<test>conținut</test>**
- Conținutul poate fi:
 - Text
 - Alte tag-uri

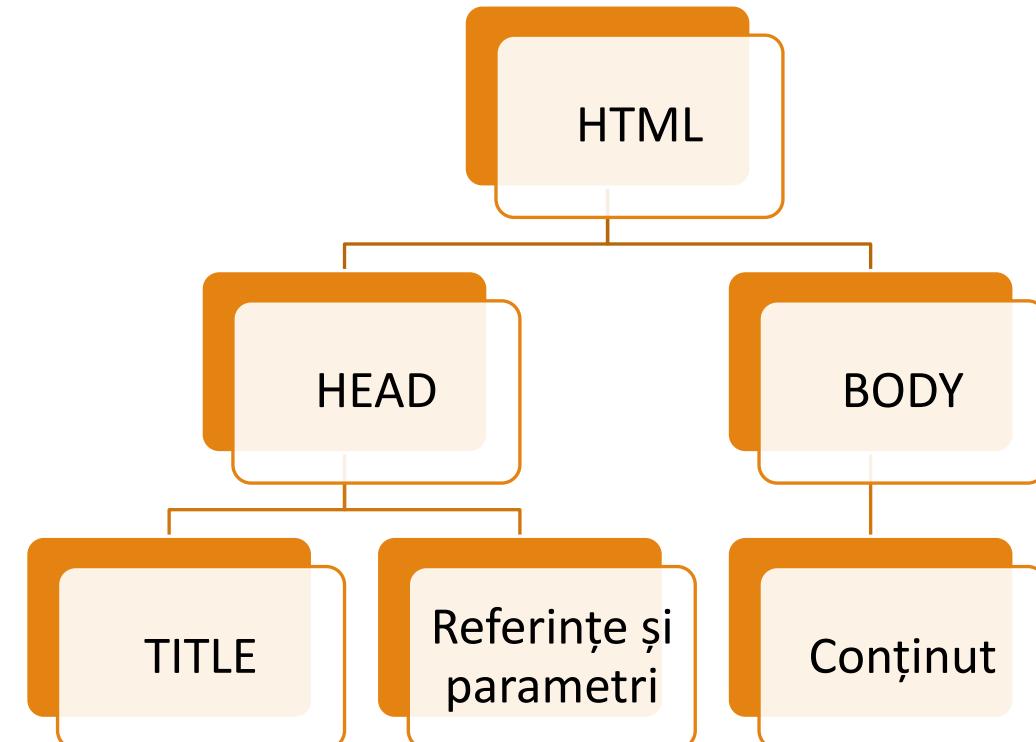
Atribut

- Perechi denumire – valoare atașate nodurilor
- **<test nume1="valoare1" nume2="valoare2">conținut</test>**

Caractere speciale: < - < | > - > | " - " | ' - ' | & - &

Structura unui document HTML

```
<!DOCTYPE html>  
  
<html>  
  <head>  
    <title>titlu</title>  
    <referințe și parametri>  
  </head>  
  <body>  
    <contenut>  
  </body>  
  
</html>
```



Tipuri de elemente

Text

- Formatare: ***h1 – h6, p, pre, div, em, strong, span, br, a***
- Liste: ***ul, ol, li***
- Tabele: ***table, thead, tbody, tfoot, tr, th, td***

Formulare: ***form, input, select, option***

Imagine raster

- *img, canvas*

Imagine vectorială: ***svg***

Audio și video: ***audio, video***

Formatare Text

H1, H2, ..., H6 – titluri

- marchează titlurile secțiunilor din document (1 este cel mai important)

P – marchează paragrafele de text

PRE – text preformatat

- Previne **comasarea spațiilor** pentru conținutul tag-ului (în tot subarborele)

DIV, SPAN – containere generice folosite pentru gruparea conținutului

STRONG, EM – folosite pentru marcarea fragmentelor mai importante într-un text

BR – inserează o linie nouă

A – creează un hyperlink (adresa destinație este specificată prin atributul **href**)

Liste și tabele

Liste

- **UL** – *unordered list* (uzual afișate cu buline)
- **OL** – *ordered list* (uzual afișate numeroate)
- Ambele conțin elemente marcate cu **LI** (*list item*)

Tabele

- **TABLE** – reprezintă un tabel și poate conține
 - Secțiuni marcate cu **THEAD**, **TBODY** și **TFOOT**
 - Un titlu marcat cu **CAPTION**; titlul poate conține orice fragment HTML și este afișat în afara tabelului
- Secțiunile conțin rânduri marcate cu **TR**
- Rândurile conțin celule marcate cu **TH** (*table header*) sau **TD** (*table data*);
 - celulele pot conține orice fragment HTML
 - O celulă se poate întinde pe mai multe rânduri / coloane prin intermediul atributelor **rowspan** și **colspan**

Formulare

Sunt conținute în interiorul unui tag **FORM** care poate avea următoarele attribute:

- **action** – adresa la care trebuie trimise datele din formular
- **method** – metoda HTTP care va fi utilizată la transmiterea cererii

Majoritatea controalelor pot fi inserate prin un tag **INPUT** cu următoarele attribute:

- **type** – determină tipul controlului (*text* – implicit, *password*, *radio*, *checkbox*, *file*, *button*, *submit*, *reset*, *hidden*)
- **name** – denumirea sub care va fi transmisă valoarea (a nu se confunda cu atributul **id**)
- **value** – conține valoarea stocată în control

Pentru combo box se folosesc elemente de tip **SELECT** (cu atribut **name**), iar definirea opțiunilor se realizează prin elemente de tip **OPTION** (cu atribut **value**)

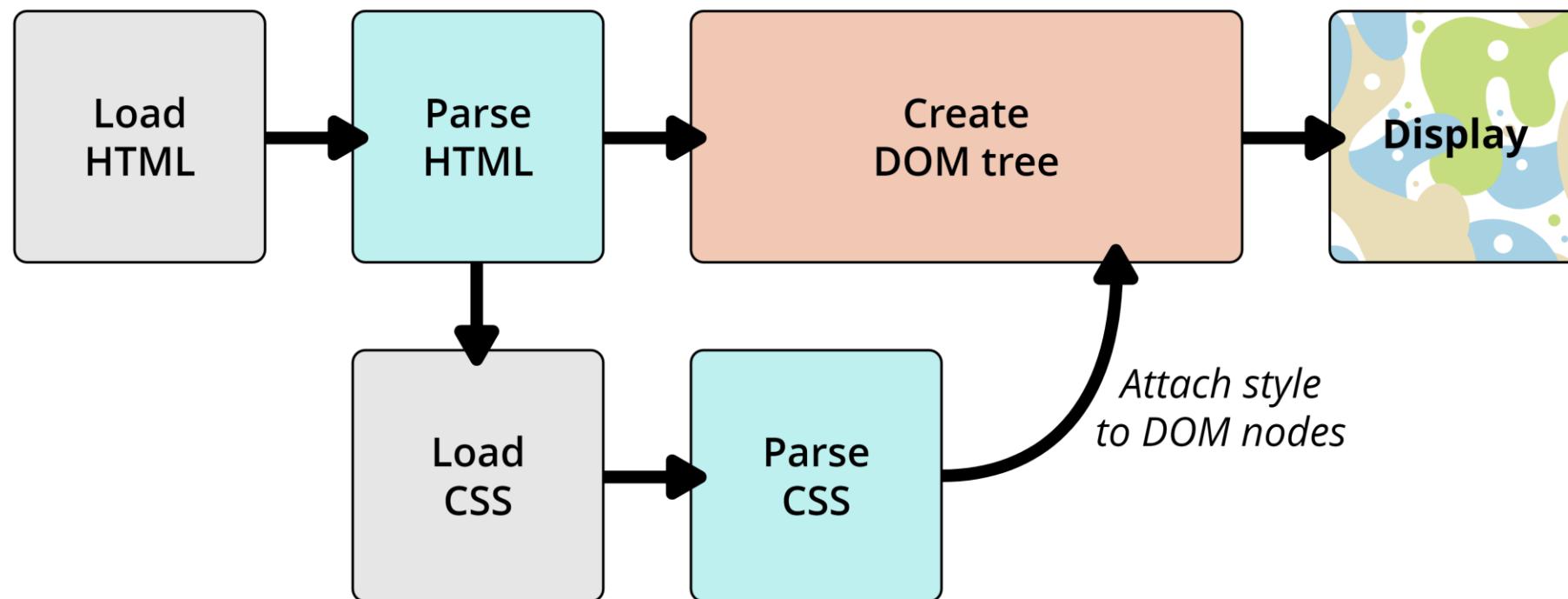
Asocierea de etichete pentru controale se realizează prin elemente de tip **LABEL** (cu atribut **for**)

Elemente de tip *block* și *inline*

	Pe linie nouă	Lățime	Înălțime	Conținut
Block	Mereu	<ul style="list-style-type: none">• 100% din lățimea părintelui• Poate fi setată	<ul style="list-style-type: none">• Înălțimea conținutului• Poate fi setată	Elemente de tip <i>block</i> și / sau <i>inline</i>
Inline	Doar dacă nu mai are loc pe linia curentă	<ul style="list-style-type: none">• Lățimea conținutului• NU poate fi modificată	<ul style="list-style-type: none">• Înălțimea conținutului• NU poate fi modificată	Doar elemente <i>inline</i>

Cascading Style Sheets

Modificarea modului de afișare a elementelor din DOM



Cascading Style Sheets

Format dintr-o serie de reguli de forma:

selector

{

proprietate 1 : valoare 1;

proprietate 2 : valoare 2;

....

}

Utilizare CSS în HTML

Tag **style** – stiluri interne

```
<style type="text/css">  
    th { font-weight: bold; }  
</style>
```

Tag **link** – stiluri externe

```
<link rel="stylesheet" type="text/css" href="test.css" />
```

Atribut **style** – stiluri inline

```
<p style="color: red; font-size: 12pt;">...</p>
```

Selectori

Identifică nodurile din arbore asupra cărora se vor opera modificările

Selectori simpli:

- Element: **TIP_ELEMENT**
- Identificator: **#ID**
- Clasă: **.NUME_CLASĂ**

Alți selectori:

- Universal: *
- Atribut: **selector[atribut="valoare"]**
- Nod copil sau descendent: **selector > selector** sau **selector selector**

Grupare: **selector, selector**

Combinare selectori: **TIP_ELEMENT#ID, TIP_ELEMENT.NUME_CLASĂ**

Reguli de aplicare

Moștenire

- Propagarea valorilor de la un nod părinte la nodurile copil
- Nu toate proprietățile sunt moștenibile

Cascadare

- Mecanismul de alegere aplicat în cazul în care avem mai multe valori în conflict
- Reguli de cascadare
 - Importanță: reguli marcate ca *!important*
 - Specificitatea selectorului
 - Ordinea în fișierul sursă

Formatare text și culori

Proprietăți:

- *font-family* – denumire font sau familie (ex: sans-serif, monospace, Arial, Tahoma, ...)
- *font-size* – dimensiune caractere (ex: small, medium, 12pt, 0.8em, ...)
- *font-weight* – grosime caractere (ex: normal, bold, 400, 700, ...)
- *font-style* – tip caractere (ex: normal, italic, oblique, ...)
- *color, background-color* – culoare text sau fundal (ex: red, #EE0000, #E00, rgb(238, 0, 0), ...)

Unități de măsură:

- relative: %, em, ...
- absolute: pt (1 / 72 inch), px (1 / 90 inch), cm,

Exemplu

```
body {font-family: tahoma; font-size: 11pt; color: blue;}
```

Setare margini și dimensiuni

Setare dimensiuni:

- *width*
- *height*
- *min/max-width/height*
- *overflow*:
 - **visible** – întreg conținutul este afișat, posibil în afara celulei
 - *hidden* - *conținutul din afara celulei nu este afișat*
 - *scroll* – *hidden + afișare bară de scroll*
 - *auto* – *afișează bara de scroll doar dacă este necesară*

Proprietățile *width* și *height* se referă doar la dimensiunea secțiunii *content* (fără *padding* / *border* / *margin*)



Setare margini și dimensiuni

Setare margini:

- padding sau margin / -top / -bottom / -left / -right
 - *padding*:
 - *3px 4px 5px 6px* (top, right, bottom, left) sau
 - *2px 3px* (top = bottom = 2px, right = left = 3px) sau
 - *3px* (toate laturile)
 - *padding-top: 5px* (setare pentru o singură latură)
 - similar pentru *margin*
 - border / -top / -bottom / -left / -right / -width / -style / -color
 - *border: 2px dotted green*
 - *border-style: solid*
 - *border-bottom-width: 3px;*



Controlul poziționării

Control layout:

- display: none / inline / block
- float: none / left / right
- clear: none / left / right / both

Poziționare în cadrul paginii

- top, right, bottom, left, z-index
 - position:
 - static: poziționare normală, proprietățile (top, ...) sunt ignorate
 - relative: poziționare normală, proprietățile (top, ...) sunt respectate
-
- absolute: nu participă la poziționarea normală; poziționat față de primul parinte cu poziționare absolută
 - fixed: nu participă la poziționarea normală; poziționat față de fereastra browser-ului

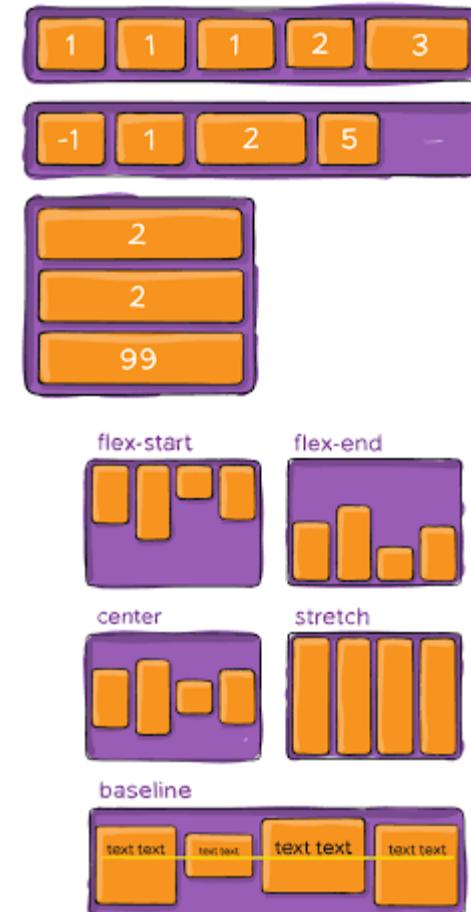
Flexbox (Optional)

Sistem de layout utilizat pentru alinierea conținutului bazat pe un **model 1D** (o singură axă principală)

- Permite poziționarea atât pe verticală (linii) cât și pe orizontală (coloane)
- Proprietăți la nivelul container-ului părinte:
 - `display: flex;`
 - `flex-direction`, `flex-wrap`, `justify-content`, `align-items`, ...
- Proprietăți la nivelul elementelor copil:
 - `order`, `flex-grow/shrink/basis`, `align-self`

Resurse:

- <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox



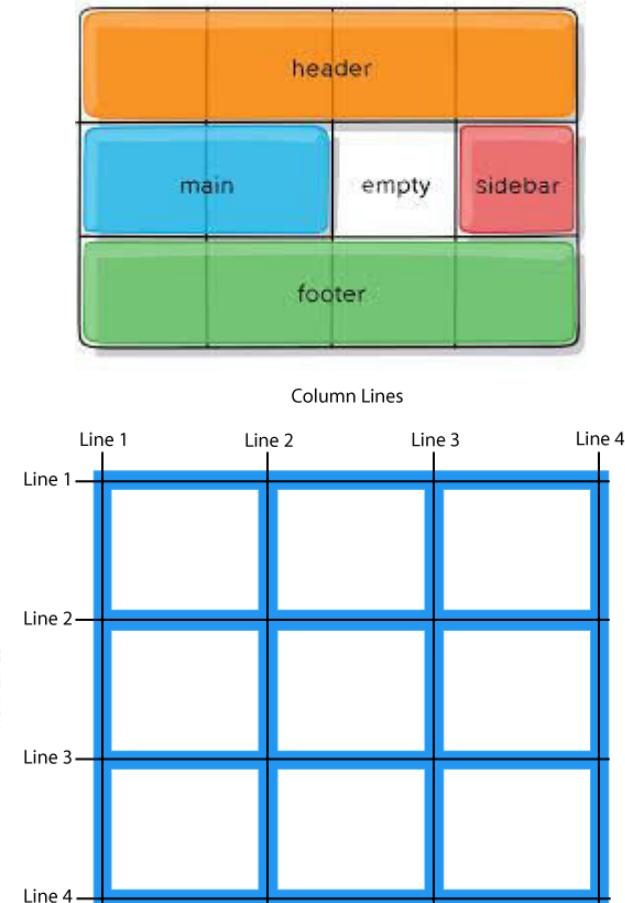
Grid (Optional)

Sistem de layout utilizat pentru alinierea conținutului bazat pe un **model 2D** (două axe principale)

- Permite poziționarea elementelor copil într-o structură matriceală
- Proprietăți la nivelul container-ului părinte:
 - **display: grid;**
 - grid-template-columns, grid-template-rows
 - grid-column/row-gap
- Proprietăți la nivelul elementelor copil:
 - grid-column-start, grid-column-end
 - grid-row-start, grid-row-end

Resurse:

- <https://css-tricks.com/snippets/css/complete-guide-grid/>
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout



Media Queries (Optional)

Sintaxă specială introdusă în CSS3 care permite aplicarea unor reguli doar în anumite condiții:

- În funcție de mediu: *all / screen / print / speech*
- În funcție de caracteristicile mediului (*width, height, ...*)

Sintaxa:

```
@media not/only mediatype and/not/only (media feature) {  
    .selector { ... }  
}
```

Resurse:

- https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries



Bootstrap (Optional)

Bibliotecă CSS

- Dezvoltată de Twitter
- Asigură uniformitatea
- Permite dezvoltarea facilă de aplicații responsive
- Sistem de layout bazat pe rânduri și coloane
- Colecție de componente
- Customizare simplă bazată pe teme

Resurse:

- <https://getbootstrap.com/docs/4.5/getting-started/introduction/>

Limbajul JavaScript

Caracteristici

- Dinamic
- Limbaj funcțional
- Bazat pe obiecte

Utilizare

- Manipulare noduri DOM
- Evenimente
- Procesare
- Comunicare la distanță

Modalități de includere în HTML

Prin intermediul tag-ului `<script>`

A) Fișier extern

```
<script type="text/javascript" src="lib/test.js"></script>
```

B) În cadrul paginii

```
<script type="text/javascript">  
// cod JavaScript ....  
</script>
```

Execuție directă în fereastra de consolă

Afișare mesaje: `console.log(valori);`

Tipuri de date

Tipuri de date de bază

- **Number:** -3.14, 6, 2
- **Boolean:** *true, false*
- **String:** “test”
- **null, undefined**
- **Object:** { nume: “Ana”, varsta:7 } – referință
- **Symbol**

Obiecte speciale

- **Function:** *function f() {...}*
- **Array:** [1, 2, “trei”]

Variabile și expresii

Declarare

- Prin atribuire valoare (**nerecomandat**): `a = 8; a = false;`
- Folosind:
 - **let**: `let b = "test"; let b;` // block scope
 - **const**: `const b = "test"` // block scope
 - **var**: `var a = 3; var b;` // function scope

Exemple:

```
a = 10; let b = "test"; b = 1.5;
```

```
const c = 7; const v = [1, 2, 3]; v[0] = 5;
```

Variabile și expresii

Scope

- Global sau local
- Function vs block based

Evaluare expresii

- Operatori similari cu C# (în plus === - egalitate strictă)

Exemple:

a += 7 a === 10 a < b && b > c

a > 10 a++ a = “Ana” + “-“ + “Maria”

Utilizare *template strings*: `Numele este \${prenume} \${nume}.`

Vectori – Obiectul Array

Inițializare: `var v = [];` sau `var v = [1, "Ion"];` sau `var v = new Array(1, 2, 3);`

Accesare elemente: `var i = v[0]; v[1] = 23;`

Dimensiune: `v.length (read / write)`

Metode:

- `push(valoare)` – adăugare la sfârșit
- `pop()` – extrage ultimul element
- `indexOf(valoare)` – întoarce poziția elementului (sau -1)
- `sort()` – sortează vectorul
- `slice(index_start, index_sfârșit)` – extrage un sub-vector
- `splice(index_start, numar_de_sters, val1, val2,)` - ștergere / înlocuire valori

Parcurgere: `for / for..of`

Functii

Declarare:

- *function suma(a,b) { return a + b; } // function declaration*
- *let suma = function(a,b) { return a + b; } // function expression*

Parametri:

- Transmiști prin valoare
- Accesibili prin *arguments*

Apel - nume_functie / expresie(parametri):

- *var rezultat = suma(7,3);*
- *var test = function(val) { return val + 1;}();*

Obiecte funcție (.apply(), .call(), .toString(), length ...)

Closure – Environment Model of Evaluation – vezi <https://sicp.comp.nus.edu.sg/chapters/52>

Arrow functions

Utilizare map / reduce / filter / find

Obiecte

Obiect = colecție de proprietăți (perechi *nume* = *valoare*)

Declarare obiecte

- Literali: `let ob = { nume: "Ana", varsta: 7 }`
- *new*: `let ob = new Object();`

Accesare proprietăți

- Citire: `let v = ob.varsta;` sau `let v = ob["varsta"];` *for (v in ob) {...}*
- Modificare / adăugare: `ob.varsta = 8;` `ob["varsta"] = 8;` `ob.clasa = 2;`
- Ștergere: `delete ob.varsta;`

Metode

- adăugare: `ob.test = function() { console.log("test");}`
- *this*: `ob.afisare = function() { console.log("Nume: " + this.nume); }`
- Apel: `ob.afisare();`

Constructori și moștenire – optional

Funcțiile JavaScript pot fi utilizate pentru construirea de obiecte

- Exemplu: *function Persoana(ume) { this.ume = nume; this.varsta = 1; }*
- Apel: *var ob2 = new Persoana("Maria");*

prototype

- Proprietate a funcției constructor
- Definește proprietățile disponibile în toate obiectele (instanțele create)
- Exemplu: *Persoana.prototype.afisare = function() { console.log("Nume: " + this.ume); };*
- Folosit și pentru implementarea moștenirii:
 - *function Student() { this.facultate = "CSIE"; }*
 - *Student.prototype = new Persoana("-");*
 - *var s = new Student(); s.afisare();*

Clase – optional

Limbajul JavaScript permite utilizarea cuvântului cheie ***class*** pentru definirea funcțiilor constructor în forma:

```
class NumeClasa {  
    constructor(parametri) {  
        this.proprietate1 = valoare;  
        this.proprietate2 = valoare;  
        ...  
    }  
    metoda1() { ... }  
    metoda2() { ... }  
    get numeProp() { ... }  
    set numeProp(value) { ... }  
}
```

Moștenirea se implementează folosind cuvântul cheie ***extends***:

```
class Derivata extends NumeClasa /* definire metode */
```

Programare execuție funcție

A) Execuție o singură dată după un interval de timp specificat

- Programare pornire:

```
var id = setTimeout(funcție, durata[, param1, param2, ...]);
```

- Oprită:

```
clearTimeout(id);
```

B) Execuție repetată la un interval de timp fix

- Programare pornire:

```
var id = setInterval(funcție, durata[, param1, param2, ...]);
```

- Oprită:

```
clearInterval(id);
```

Observație: duratele sunt exprimate în milisecunde

DOM - Structura

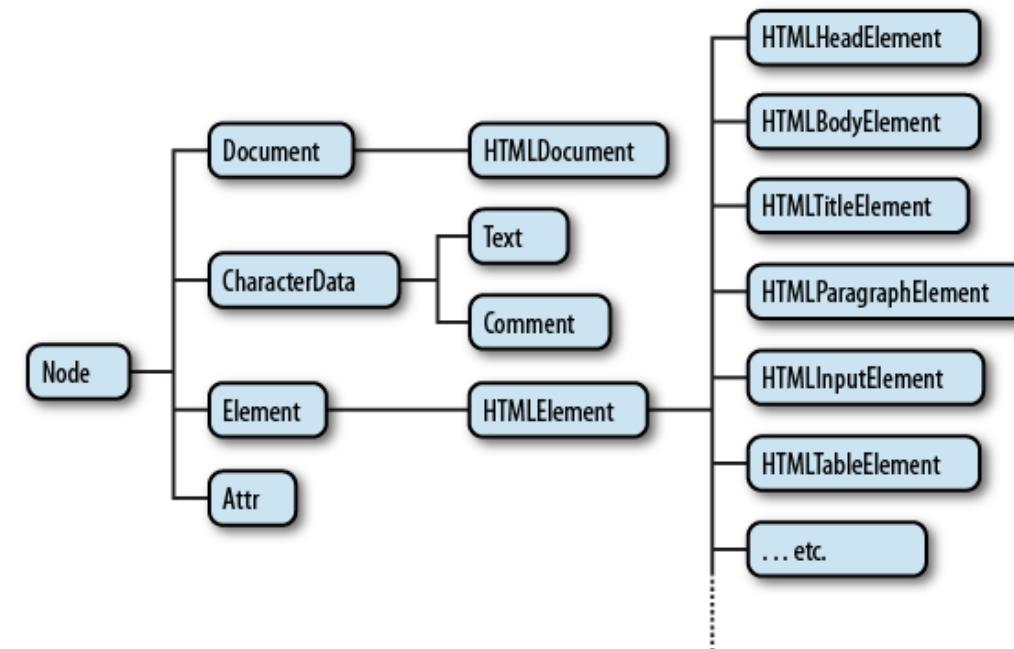
Arbore construit din obiecte JavaScript

Fiecare nod:

- Este un obiect derivat din *Node*
- Conține o colecție de referințe către nodurile copil: *childNodes / children*
- Conține referințe către nodul părinte (*parentNode*) și nodul următor (*nextSibling*)
- Conține metode pentru manipularea nodurilor copil: *append(nod), remove(), ...*

Rădăcina: *document.documentElement, document.head, document.body*

Nodurile au metode și proprietăți specifice în funcție de tag-ul HTML utilizat pentru construirea nodului



Sursa: <http://web.stanford.edu/class/cs98si/slides/the-document-object-model.html>

Regăsire noduri

Pe bază de identificator

- elem = *document.getElementById(identificator)*

Pe bază de selector CSS

- elem = element.*querySelector*("selector CSS");
- lista = element.*querySelectorAll*("selector CSS");

Alte metode

- lista = element.getElementsByClassName("clasa CSS");
- lista = element.getElementsByTagName("tag HTML");

Manipulare noduri

Construire nod:

- elem = *document.createElement("tag HTML"); / document.createTextNode("text")*
- Proprietăți: innerText, innerHTML
- Metode de adăugare nod:
 - parinte.prepend(copil)
 - parinte.append(copil)
 - nod.before(nodNou)
 - nod.after(nodNou)
 - nod.replaceWith(nodNou)
- Metode de eliminare nod:
 - parinte.removeChild(copil)
 - nod.remove()

Manipulare Atribute

Accesare atribute:

- elem.getAttribute – accesare valoare atribut individual
- elem.attributes – colecția de atribute
- elem.hasAttribute(name) – verificare existență
- elem.getAttribute(name) – obținere valoare
- elem.setAttribute(name, value) – modifica valoare
- elem.removeAttribute(name) – ștergere atribut

Asociere date proprii:

- Atribute HTML: <tag **data-denumire**=“valoare”></tag>
- Din JavaScript: element.**dataset**.denumire

Manipulare Noduri

Accesare atribute CSS:

- elem.style.numeAtributCSS
- elem.className – string, corespunzător atributului *class* din HTML
- elem.classList – obiect care permite manipularea listei de clase
 - (metode *add / remove / toggle / contains*)

Obținere coordonate element fata de fereastră: elem.getBoundingClientRect()

Exemplu:

```
let p = document.createElement("p");
p.innerText = "test";
document.querySelector("body").append(p);
p.style.backgroundColor = "red";
```

Tratare evenimente

Adăugare event handler:

- Prin attribute HTML:
 - <element atributEveniment="nume funcție">...</element>
 - Exemplu: <a **onclick="test()"**
- Prin proprietăți nod:
 - element.proprietateEveniment = funcție;
 - Exemplu: document.getElementById("test").**onclick** = function() {console.log("un mesaj");}
- **Prin metoda addEventListener:**
 - element.**addEventListener**(tip, funcție);
 - Exemplu: document.getElementById("test").**addEventListener**("click", function() {console.log("un mesaj");});

Eliminare event handler:

- element.**removeEventListener**(tip, funcție);

Tratare evenimente

Accesare element sursă – *this*

Parametri eveniment – obiect *event*

- General: *target*, *currentTarget*, *type*, *preventDefault()*
- Tastatură: *key*, *keyCode*, *altKey*, *ctrlKey*, *shiftKey*
- Mouse: *pageX*, *pageY*, *button*, *altKey*, *ctrlKey*, *shiftKey*

Evenimente

- General: *load* (*window*), *DOMContentLoaded* (*document*)
- Tastatură: *keydown*, *keypress*, *keyup*
- Mouse: *mouseenter*, *mouseleave*, *mousemove*, *mouseup*, *mousedown*, *click*, *dblclick*

Comunicarea cu serverul - optional

Prin obiecte de tip ***XMLHttpRequest***

Permite crearea, transmiterea și receptia de cereri HTTP(S)

Construire obiect cerere:

```
var cerere = new XMLHttpRequest();
```

Funcții:

- *open(method, url)* – initializează un obiect ***XMLHttpRequest***
- *send(data)* – începe procesarea asincronă a cererii
- *abort()* – anulează o cerere în desfășurare
- *setRequestHeader(header, value)* – permite transmiterea de valori în header-ul cererii
- *getResponseHeader(header)* – permite citirea valorilor din header-ele primite de la server

Comunicarea cu serverul – optional

Proprietăți:

- *readyState* – intreg care indică starea curentă a cererii (0 – netrimisă, ..., 4 – done)
- *response* – răspunsul primit de la server de tipul indicat de *responseType*
- *status, statusText* – codul HTTP pentru răspuns în format numeric și text

Evenimente - addEventListener:

- *load* – cererea a fost executată și a fost primit răspunsul
- *readystatechange* – s-a modificat starea curentă a obiectului
- *abort* – cererea a fost anulată de client
- *timeout* – a expirat timpul alocat
- *error* – cererea s-a încheiat cu o eroare

Communicarea cu serverul – Fetch API

Obiecte ***Promise*** – încapsulează o operație asincronă și callback-urile asociate

Funcție: **fetch(URL[, {options}])** – întoarce un obiect de tip *Promise*

Opțiuni:

- *method* – ‘GET’ (implicit) sau ‘POST’
- *headers* – obiect care conține headerele de trimis către server sub formă de proprietăți
- *body* – conținutul pentru cereri de tip POST (string, FormData, ...)

Obiectul răspuns:

- proprietăți pentru determinarea rezultatului (*ok*, *status*, *statusText*)
- *headers* – obiect care permite citirea datelor din header-ul HTTP
- Metode pentru citirea conținutului (*text()*, *json()*, *formData()*, ...) – întorc un obiect de tip *Promise*

JavaScript – Execuție asincronă

Instrucțiuni dedicate pentru obiecte *Promise*:

- ***async function***: declară o funcție asincronă care întoarce un obiect *Promise*
 - permite utilizarea operatorului *await* în corpul funcției
- ***await***: operator care permite execuția unei funcții asincrone ce întoarce un obiect *Promise* și continuarea execuției doar la primirea rezultatului

Exemplu:

```
async function app() {  
    var raspuns = await fetch('./dateEU.json');  
    var json = await raspuns.json();  
    console.log(json);  
}  
app();
```

Resurse:

- <http://javascript.info/async>
- <https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/fetch>

Biblioteca jQuery – optional

Colecție de funcții JavaScript pentru:

- Regăsire elemente din DOM
- Manipulare elemente, atribute și proprietăți CSS
- Înlănțuire operații pe seturi elemente
- Animație
- Comunicare HTTP

Funcția jQuery / \$ – optional

1. Regăsire elemente DOM: **`$("selector CSS")`**

- `var leg = $("a.test");`

2. Construire obiect jQuery: **`$(element)`**

- `var a = document.getElementById("leg1");`
- `var ajQuery = $(a);`

3. Construire elemente noi: **`$("cod HTML")`**

- `var p = $("<p>paragraf nou</p>");`

4. Execuție cod după inițializarea DOM: **`$(funcție)`**

- `$(function(){`
- `// operații care utilizează DOM`
- `})`

Obiecte jQuery – optional

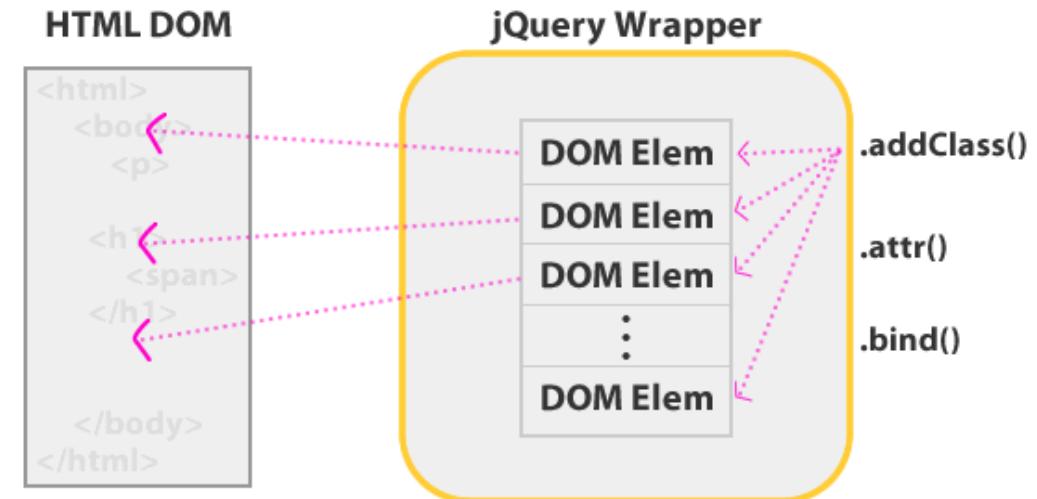
Fiecare obiect jQuery conține o colecție de 0+ referințe la obiectele DOM selectate.

Accesarea elementelor DOM:

- Operator `[]` sau funcția `get`:
 - `$("td")[3]` sau `$("td").get(3)`
 - Rezultatul obținut este un obiect DOM
- Proprietate `length`:
 - `$("td").length`

Operații asupra obiectului jQuery:

- În general se aplică pe toate elementele și întorc o referință la obiectul jQuery (pentru înlățuire)
- Funcțiile care citesc o valoare se aplică asupra primului element și întorc valoarea citită



jQuery - exemple de operații – optional

Citire / modificare atribută HTML

- `$("#idTest").attr("href", "doc.html");`
- `var url = $("#idTest").attr("href");`
- `.text() / .val()`

Citire / modificare proprietăți CSS

- `$("#idTest").css("color", "red");`

Manipulare noduri DOM

- `$("#idTest").append($("#<p>test</p>"));`
- `părinte.empty() / copil.appendTo(părinte) / copil.remove()`

Tratare evenimente

- `$("#idTest").click(func);`
- `$.on("ev", func);`

jQuery – comunicare – optional

Obținerea de date de la server prin comenzi **GET**:

\$.get(url, func) sau **\$.getJSON(url, func)**

```
$.get("date.txt", function(txt) { console.log(txt); });
```

```
$.getJSON("date.txt", function(obj) { console.log(obj.proprietate); });
```

Trimiterea de date către server prin comenzi **POST**:

\$.post(url, date[, func])

```
$.post("procesare.aspx", JSON.stringify(obj), function(obj) { console.log(obj); });
```

Opțiuni complete: **\$.ajax(...)**

II. Imagine

1. Modele de culoare

2. Grafică raster

3. Grafică vectorială

4. Animație

1. Modele de culoare

Model de culoare

- Model matematic care descrie modalitatea de reprezentare a culorilor sub formă de tupluri numerice
- Exemple: RGB, CMY

Spațiu de culori

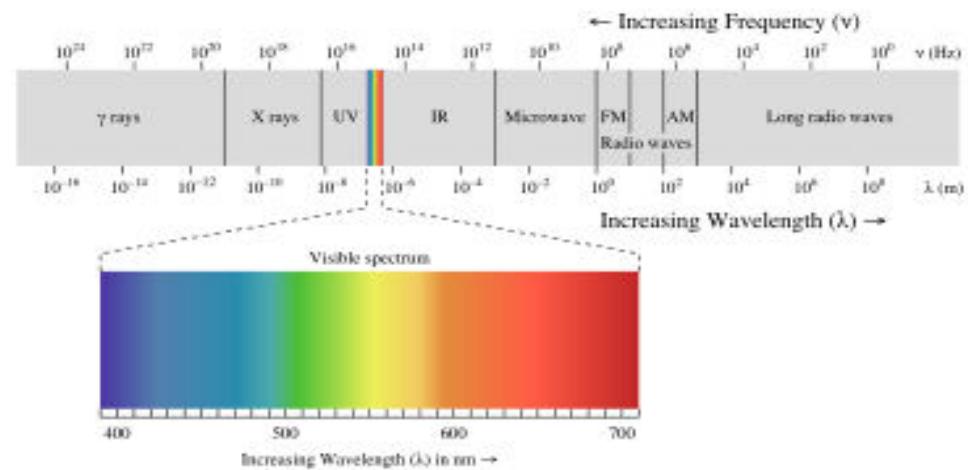
- Modelul de culoare împreună cu instrucțiunile de reprezentare fizică
- Exemple: sRGB, AdobeRGB, Pantone

Caracteristici

- Tin seama de modalitatea de percepere a luminii de către ochiul uman
- Culori de bază albastru (S), verde (M) și roșu (L)

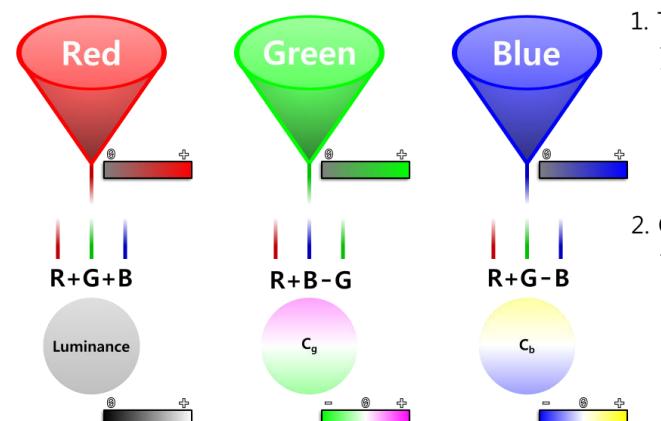
Lumina

Radiație electromagnetică cu lungimea de undă aproximativ între 400 și 700 nm



Vederea umană:

- Trei tipuri de celule conice fotosensibile (pentru culorile roșu, verde, albastru)
- Culoarea este procesată pe baza a trei componente (luminanță, C_g și C_b)



1. Trichromatic Stage

Trichromatic cone cells respond positively to one of three frequencies exhibited by photons arriving on their surface.

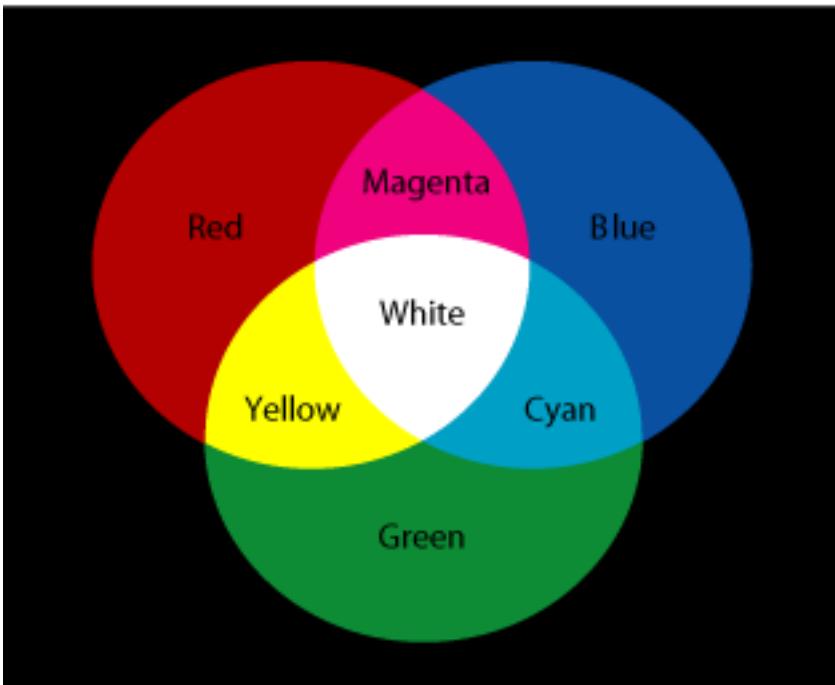
2. Opponent Process Stage

The three color channels are discovered by nearby opponent cells.

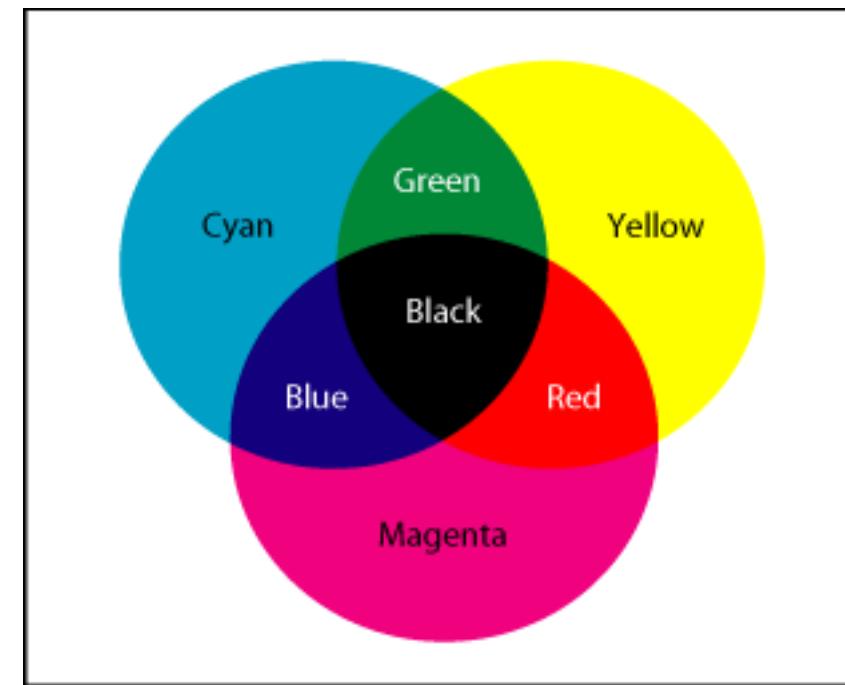
Opponent cells tuned to luminosity are excited by the red, green, and blue color signals.

C_g cells are excited by red and blue and inhibited by green. C_b cells are excited by blue and inhibited by red and green.

Tipuri de modele



Model aditiv



Model substractiv

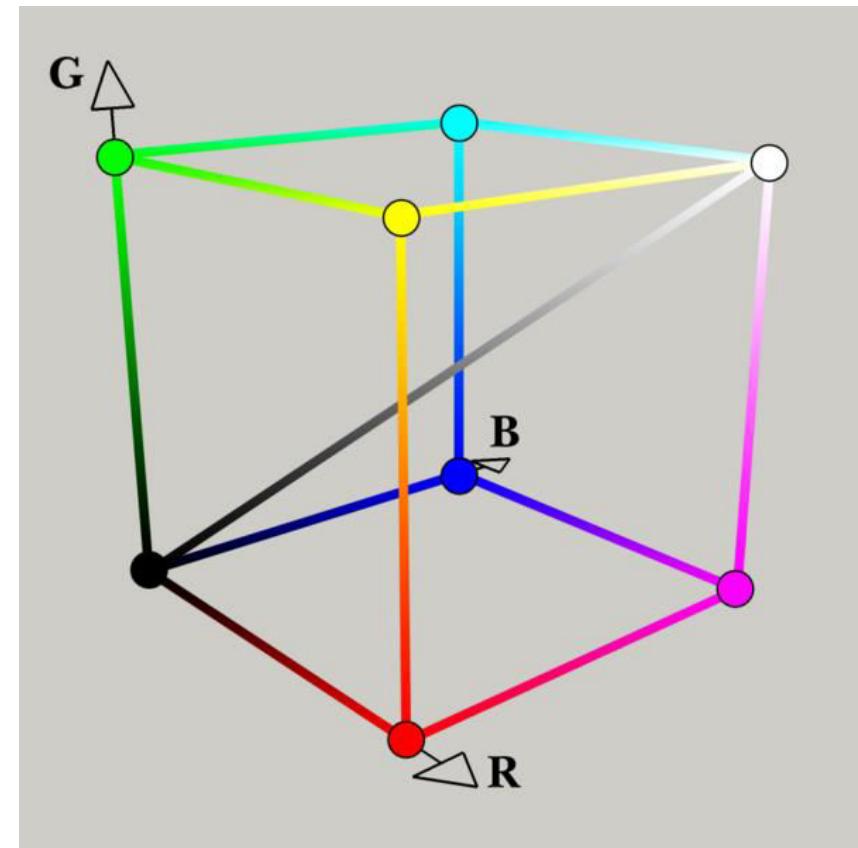
Modelul RGB

Model aditiv

Bazat pe culorile de bază roșu, verde și albastru

Culoarea variază în funcție de dispozitiv

(în lipsa unui spațiu de culoare)



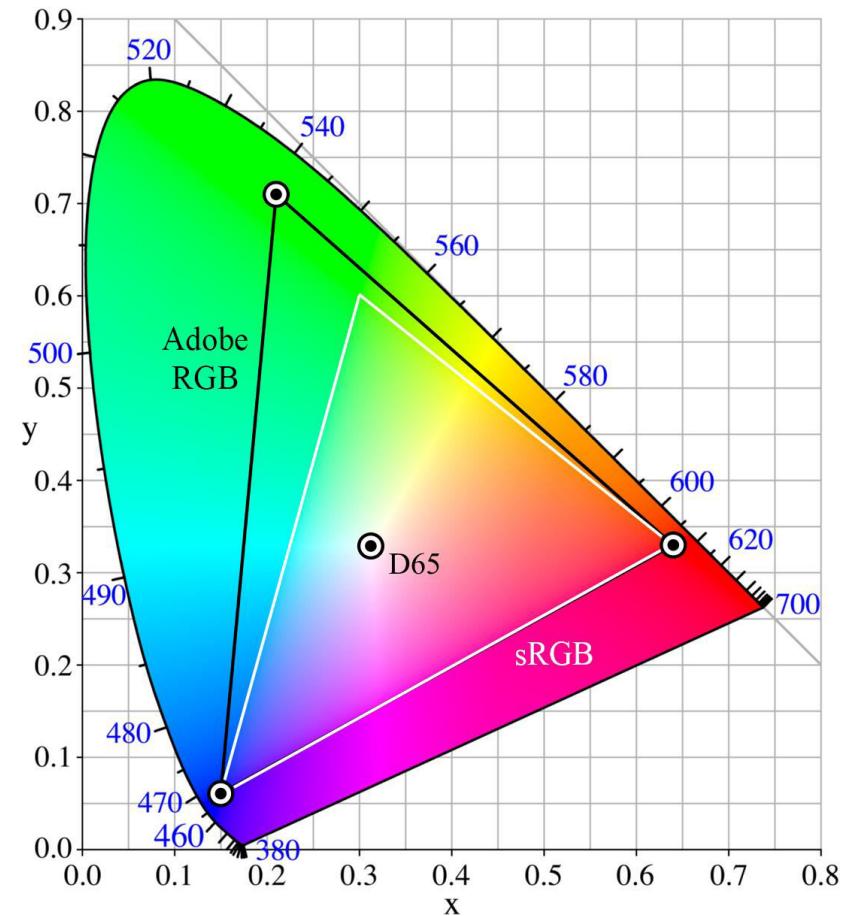
Spații de culoare RGB

sRGB

- Dezvoltat de HP și Microsoft
- Standard pentru monitoare / imprimante / web
- Utilizat ca standard implicit

AdobeRGB

- Dezvoltat de Adobe
- Acoperă aproape complet spațiul de culori CMYK



Modelul HSL (B,V)

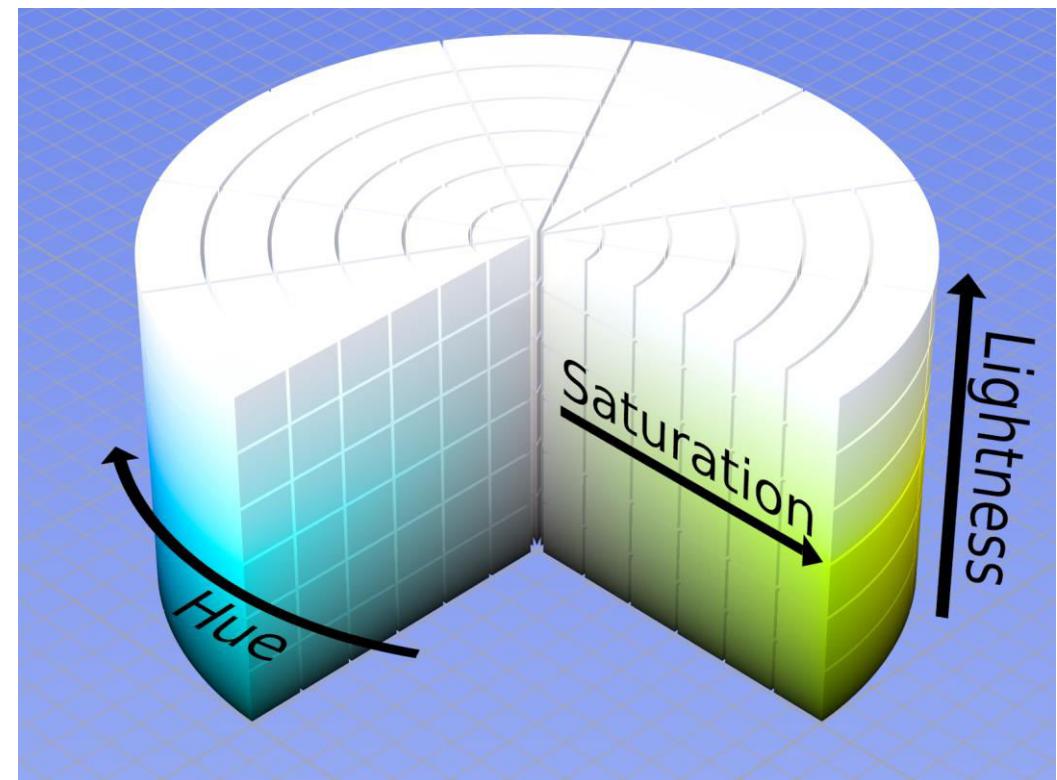
Reprezentare sub formă de coordonate cilindrice

Bazat pe aceleași culori de bază:

- Red - 0^0
- Green - 120^0
- Blue - 240^0

Axa centrală – tonuri de gri

Utilizat în special pentru selecție de culoare



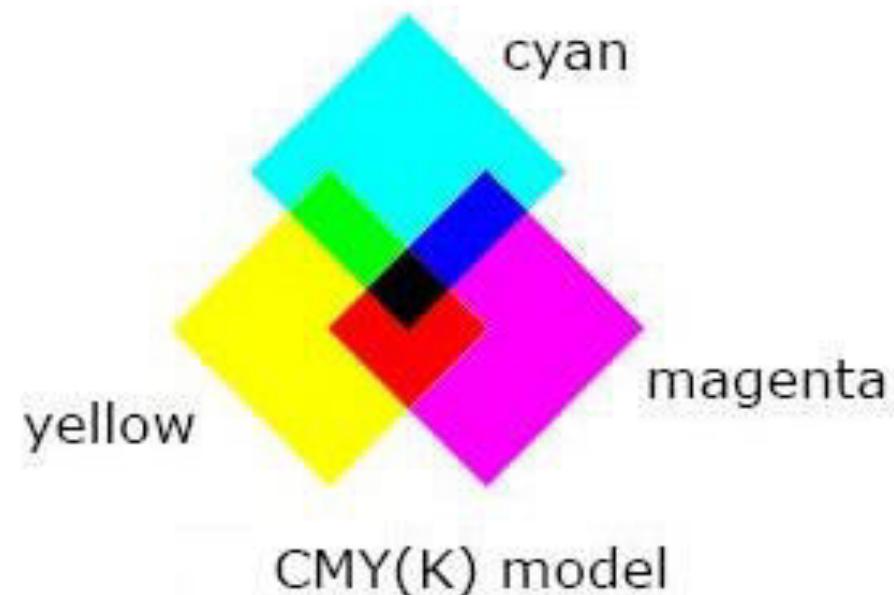
Modelul CMY(K)

Model subtractiv

Culori utilizate: cyan, magenta, yellow

Maschează culorile pe o suprafață albă

Utilizare: materiale tipărite



CSS – Specificarea colorilor

Format hexazecimal:

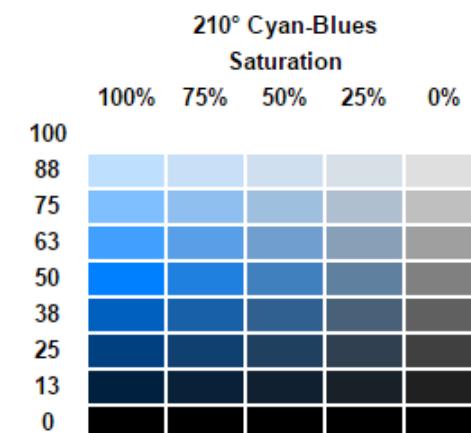
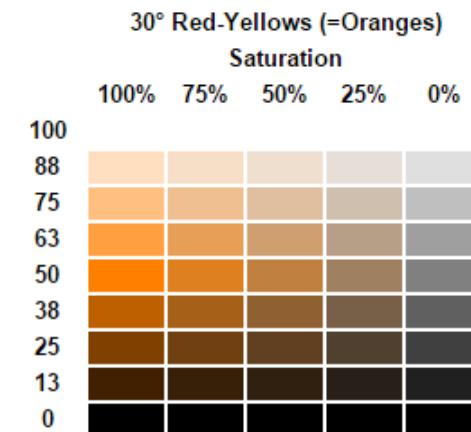
- **#rgb** sau **#rrggbb**
- **r,g,b** sunt cifre în baza 16

Format RGB

- **rgb(red, green, blue)** sau **rgba(red, green, blue, alpha)**
- **red, green, blue** – numere de la 0 la 255 sau procente
- **alpha** – număr între 0 – transparent și 1 – opac sau procent

Format HSL

- **hsl(hue, saturation, lightness)** sau **hsla(hue, saturation, lightness, alpha)**
- **hue** – număr de la 0 la 360
- **saturation, lightness** – procente
- **alpha** – număr între 0 – transparent și 1 – opac sau procent



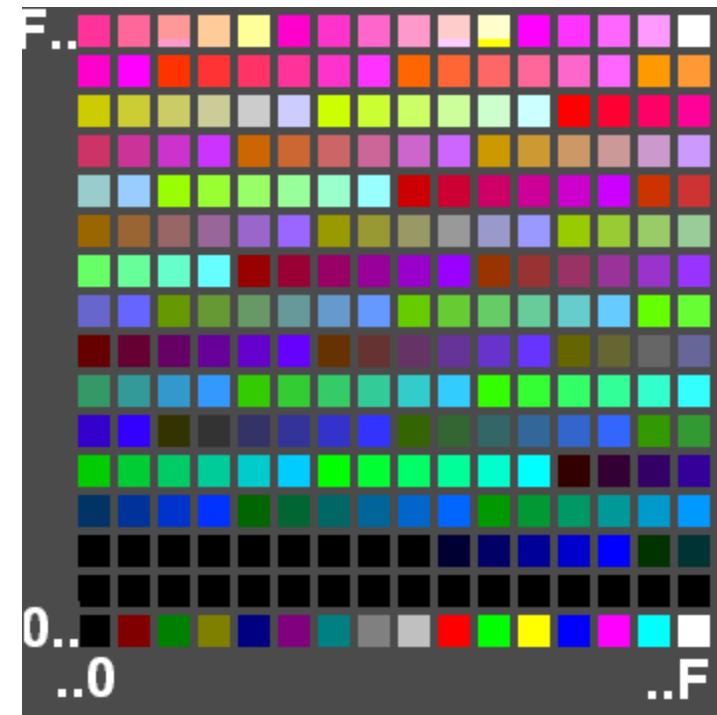
Palete de culori

Tabel de corespondență

index – tuplu (R, G, B)

Utilizare

- Reducerea cantității de informație necesară pentru reprezentarea culorilor
- Pretabilă pentru imagini cu un număr redus de culori (exemplu: diagrame fără gradienți)



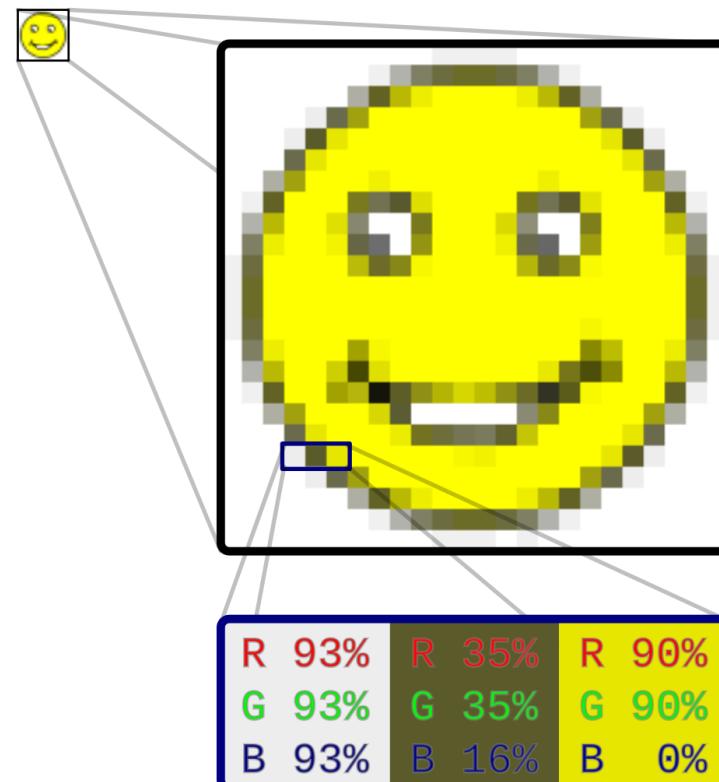
2. Grafică raster

Tipuri principale de grafică:

- raster și vectorial

Grafica raster (bitmap, matriceală)

- Reprezentare sub formă de matrice de puncte
- Fiecare punct (denumit pixel) stochează informația de culoare
- Culorile sunt stocate conform unui model de culoare
 - direct
 - prin intermediul unei palete de culori



Sursa: <http://commons.wikimedia.org/wiki/File:Rgb-raster-image.svg>

Informații generale

Caracteristici principale imagine raster

- Rezoluție (numărul de linii și coloane stocate în matrice, numărul total de pixeli sau densitate)
- Adâncime de culoare (cantitatea de informație stocată de către fiecare pixel)

Utilizare

- Reprezentare imagine pe monitor
- Captare imagini din surse externe

Avantaje și dezavantaje

- + Poate reprezenta orice imagine
- Codaj sărac în informație (nu ia în considerare semantica imaginii)
- Dimensiune mare
- Nu se pot adapta unei scări variabile de vizualizare

Desenare elemente grafice

Presupune colorarea celulelor matricei plecând de la ecuația matematică

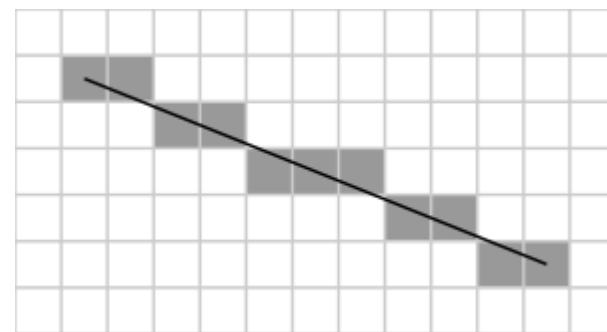
Exemplu: segment de dreaptă $(x_1, y_1) \rightarrow (x_2, y_2)$

Se pleacă de la ecuația dreptei $y=mx+b$ și se determină valorile pentru $m=(y_2-y_1)/(x_2-x_1)$ și $b=(y_1 - mx_1)$

Se colorează punctele corespunzătoare ecuației:

```
dx = x2 - x1; dy = y2 - y1;
for x from x1 to x2 {
    y = y1 + dy * (x - x1) / dx
    plot(x, y)
}
```

(se presupun $d_x \geq d_y$ și $x_1 > x_2$)



Elementul canvas

HTML – includere în document:

```
<canvas id="test" width="250" height="150"></canvas>
```

JavaScript – obținere referință context grafic:

```
// obținere obiect HTMLCanvasElement din DOM  
var canvas = document.getElementById('test'); // sau var canvas = $('#test')[0];  
  
var w = canvas.width, h = canvas.height;  
  
// obținere context grafic (obiect de tip CanvasRenderingContext2D)  
var ctx = canvas.getContext('2d');
```

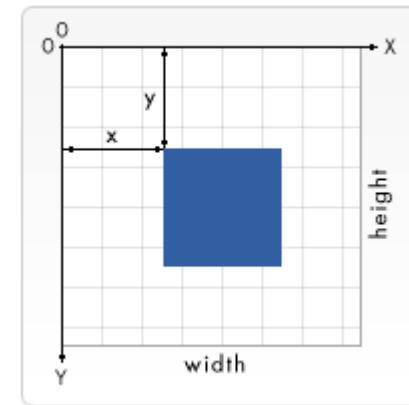
canvas – Modalități de desenare

Desenare directă – dreptunghiuri:

- *fillRect(x, y, width, height)* – umplere suprafață
- *strokeRect(x, y, width, height)* – desenare contur
- *clearRect(x, y, width, height)* – ștergere suprafață

Desenare folosind căi:

- ***beginPath()*** – deschide o cale nouă
- instrucțiuni de desenare / poziționare
- opțional *closePath()* – închide calea (unește cu punctul de început)
- ***fill()*** – umple calea și / sau ***stroke()*** – desenează liniile



canvas – Instrucțiuni desenare

Deplasare:

moveTo(x, y) – modifică poziția curentă

Desenare:

lineTo(x, y) – adaugă o linie de la poziția curentă la punctul specificat

rect(x, y, width, height) – adaugă un dreptunghi

arc(x, y, radius, startAngle, endAngle, anticlockwise) – adaugă un arc de cerc

quadraticCurveTo(cp1x, cp1y, x, y) – adaugă o curbă quadratică (un punct de control)

bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y) – adaugă o curbă Bézier (două puncte de control)

Desenare text: *fillText* sau *strokeText(string, x, y)*

canvas – Atribute

Salvare și restaurare atribute context folosind stiva: ***save()*** și ***restore()***

Atribute:

- *fillStyle* sau *strokeStyle* = “culoare” – modifică culoarea de desenare
 - `ctx.fillStyle = "#FFA500";`
 - `ctx.strokeStyle = "rgba(255,165,0,1);`
- *lineWidth* = dimensiune – modifică grosimea liniilor
- *lineCap*, *lineJoin*, *miterLimit* – modifică proprietățile liniilor (rotunjire capete, îmbinări)
- *font* = “specificație font” – modifică caracteristicile textului (exemplu: “bold 18px Arial”)
- *textAlign* – poziția textului față de coordonata x (*left*, *right* sau *center*)
- *textBaseline* – poziția textului față de coordonata y (*top*, *hanging*, *middle*, *alphabetic* sau *bottom*)

canvas – Desenare curbe

Suport pentru desenare curbe *Bézier*

- cuadratică (un punct de control)
- cubică (două puncte de control)

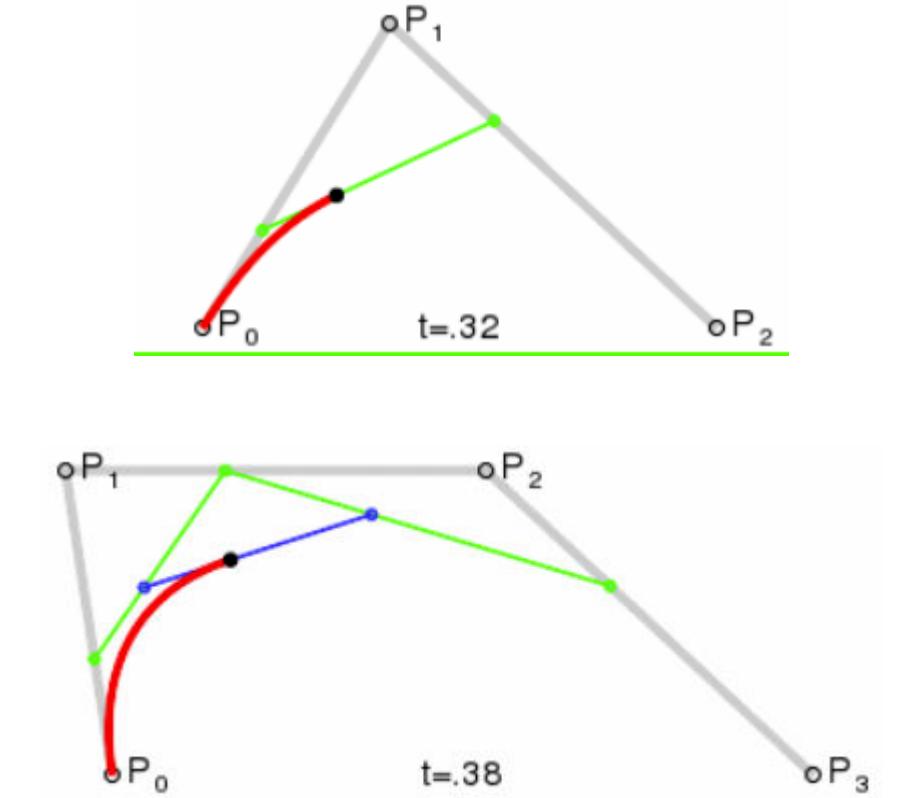
Formule:

$$P = (1-t)P_1 + tP_2$$

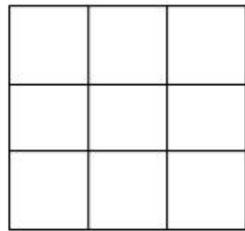
$$P = (1-t)^2P_1 + 2(1-t)P_2 + t^2P_3$$

Algoritmul de *Casteljau* (cuadratică):

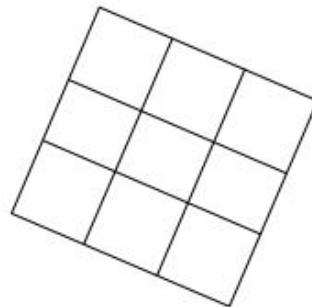
- se consideră punctele P_0 , P_1 și P_2
- se construiesc segmentele P_0P_1 și P_1P_2
- Pentru t de la 0 la 1
 - Se determină punctul aflat la o distanță proporțională t față de începutul segmentului pentru P_0P_1 și P_1P_2
 - Se construiește un segment nou din cele două puncte obținute și se repetă procesul



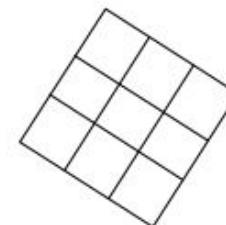
Tipuri de transformări 2D



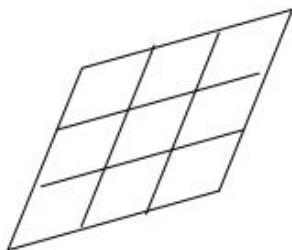
Identică



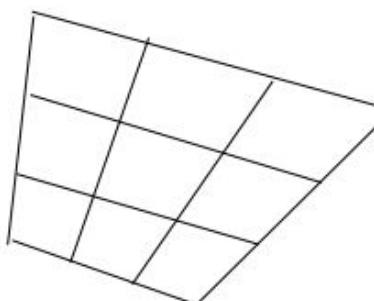
Euclidiană
(translație, rotație)



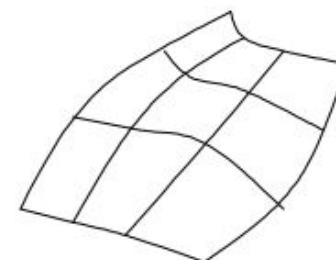
Helmert
(+scalare)



Afină
(+deformare)



Proiectivă
(+paralelism)

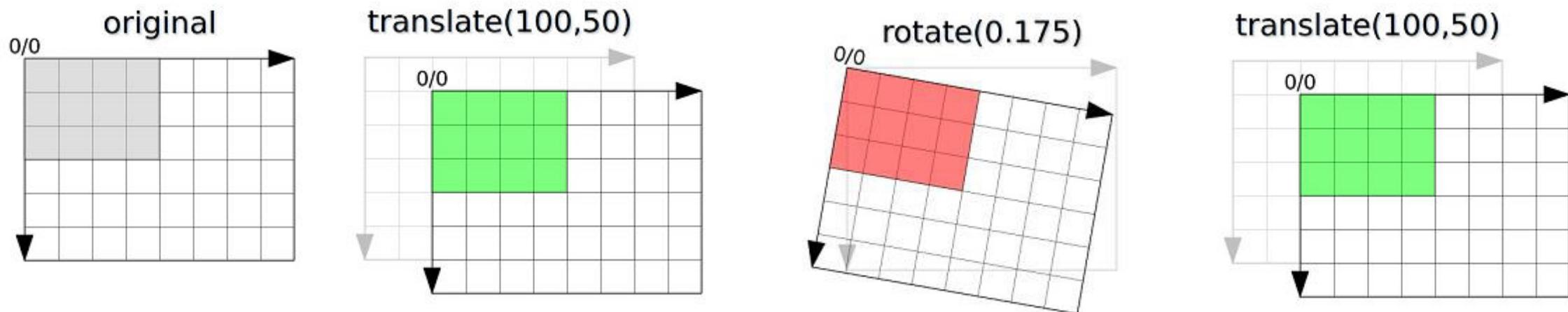


Polinomială
(+proprietăți geometrice)

canvas – Transformări

Operații de transformare simplă:

- *translate(x, y)* – translatează sistemul de coordonate cu numărul de pixeli specificați
- *rotate(angle)* – rotește sistemul de coordonate cu unghiul specificat (în radiani)
- *scale(x, y)* – scalează sistemul de coordonate cu factorii specificați (în [0...1])



canvas - Transformări

Forma generală pentru transformări afine:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Componere transformare (prin înmulțire):

transform(m11, m12, m21, m22, dx, dy)

Înlocuire transformare curentă:

setTransform(m11, m12, m21, m22, dx, dy)

resetTransform() – revenire la sistemul de coordonate standard

canvas - Transformări

Semnificație parametri:

- m_{11} : scalare pentru axa Ox
- m_{12} : rotire pentru axa Ox
- m_{21} : rotire pentru axa Oy
- m_{22} : scalare pentru axa Oy
- d_x : translatare pentru axa Ox
- d_y : translatare pentru axa Oy

Imaginea raster în context Web

Afișarea imaginilor în HTML:

- Utilizare nod DOM (HTMLImageElement)
 - *width, height*: dimensiunea obiectului în fereastră
 - *naturalWidth, naturalHeight*: dimensiunea matricei raster
 - *src*: URL-ul imaginii sursă; modificarea determină începutul procesului de încărcare (asincron)
 - eveniment *load*
- Exemplu:

```
var image = $("<img>")
    .on('load', function () { $("body").append($(this)); })
    .attr("src", "media/Penguins.jpg");
```

- Alte metode:
 - Element *input* cu *type='file'*
 - *Drag and drop*

canvas – Desenare imagini

Desenare fără scalare:

drawImage(image, x, y)

Desenare cu preluare porțiune imagine și scalare:

drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)

Surse posibile:

- element **img**
- alt element **canvas**
- element **video**

canvas – Acces raster

Stocare raster – obiect *ImageData*:

- *width, height*: dimensiunile matricei
- *data*: vector care conține matricea liniarizată

Operații:

- *getImageData(left, top, width, height)*: extrage o porțiune din imagine ca obiect *ImageData*
- *putImageData(imageData, x, y)*: aplică datele pe imaginea afișată în *canvas*

Restricții de origine

canvas – Access raster

Parcure elemente:

```
var imageData = context.getImageData(  
    0, 0, canvas.width, canvas.height);  
  
for (var y = 0; y < canvas.height; y++) {  
    for (var x = 0; x < canvas.width; x++) {  
        var i = (y * canvas.width * 4) + x * 4;  
  
        var rosu = imageData.data[i]; // [0..255]  
        var verde = imageData.data[i+1]; // [0..255]  
        var albastru = imageData.data[i+2]; // [0..255]  
        var transparenta = imageData.data[i+3]; // [0..255]  
    }  
}
```

Efecte simple de culoare

Filtrare culoare

- Exemplu pentru roșu: $r' = r; g = 0; b = 0$

Negativ

- $r' = 255 - r; g' = 255 - g; b' = 255 - b;$

Transformare în tonuri de gri

- $r' = g' = b' = 0.299 * r + 0.587 * g + 0.114 * b$

Modificare strălucire

- $r' = r + \text{valoare}; \text{if } (r > 255) r = 255 \text{ else if } (r < 0) r = 0;$
- similar pentru celelalte două componente

Modificare contrast

- $r' = (((r / 255) - 0.5) * \text{valoare} + 0.5) * 255; \text{if } (r > 255) r = 255 \text{ else if } (r < 0) r = 0;$
- similar pentru celelalte două componente

Transformare RGB → HSL

$$M = \max(R, G, B)$$

$$m = \min(R, G, B)$$

$$C = M - m$$

$$H' = \begin{cases} \text{undefined}, & \text{if } C = 0 \\ \frac{G-B}{C} \bmod 6, & \text{if } M = R \\ \frac{B-R}{C} + 2, & \text{if } M = G \\ \frac{R-G}{C} + 4, & \text{if } M = B \end{cases}$$

$$H = 60^\circ \times H'$$

$$L = \frac{1}{2}(M + m) = 0.5 * \max(R, G, B) + 0.5 * \min(R, G, B)$$

$$S_{HSL} = \begin{cases} 0, & \text{if } L \in \{0, 1\} \\ \frac{C}{1-|2L-1|}, & \text{otherwise} \end{cases}$$

Transformare HSL → RGB

$$C = (1 - |2L - 1|) \times S_{HSL}$$

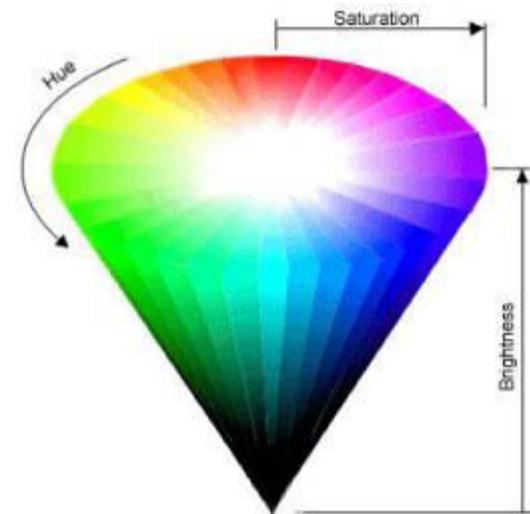
$$H' = \frac{H}{60^\circ}$$

$$X = C(1 - |H' \bmod 2 - 1|)$$

$$(R_1, G_1, B_1) = \begin{cases} (0, 0, 0) & \text{if } H \text{ is undefined} \\ (C, X, 0) & \text{if } 0 \leq H' < 1 \\ (X, C, 0) & \text{if } 1 \leq H' < 2 \\ (0, C, X) & \text{if } 2 \leq H' < 3 \\ (0, X, C) & \text{if } 3 \leq H' < 4 \\ (X, 0, C) & \text{if } 4 \leq H' < 5 \\ (C, 0, X) & \text{if } 5 \leq H' < 6 \end{cases}$$

$$m = L - \frac{1}{2}C$$

$$(R, G, B) = (R_1 + m, G_1 + m, B_1 + m)$$



Histograma unei imagini

Permite analiza distribuției tonurilor în cadrul unei imagini

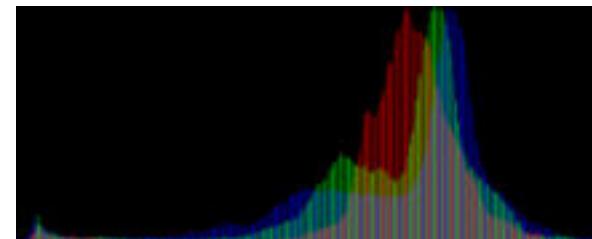
Construcție:

- Ox: intensitatea (de obicei 0-255)
- Oy: numărul de pixeli existenți pentru fiecare valoare a intensității

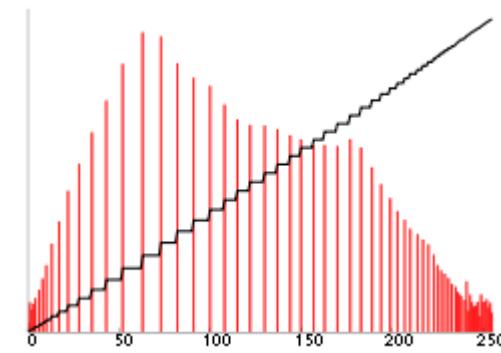
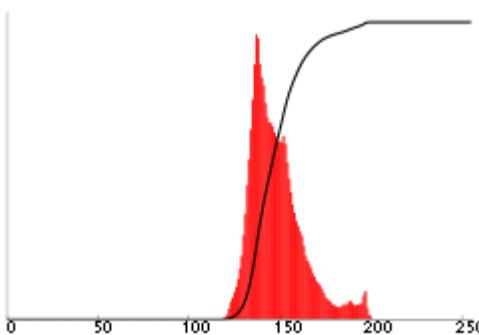
Poate fi construită pentru fiecare canal în parte

Exemple de utilizări:

- Îmbunătățire contrast prin egalizare histogramă
- Descoperire muchii / segmentare imagine

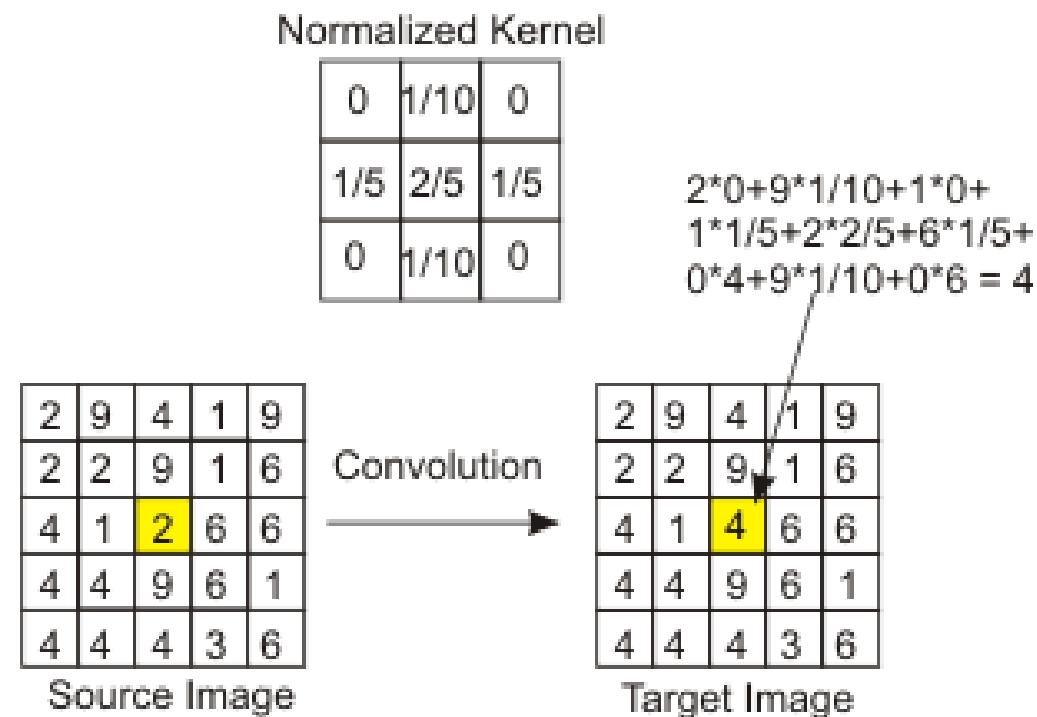


Exemplu egalizare histogramă



Filtre de conoluție

Calculează valoarea fiecărui pixel în funcție de valorile pixelilor alăturați



Filtre de conoluție

Algoritm general:

pentru fiecare valoare $v(x,y)$ din matricea originală

acumulator = 0

pentru fiecare valoare $k(i,j)$ din matricea de conoluție

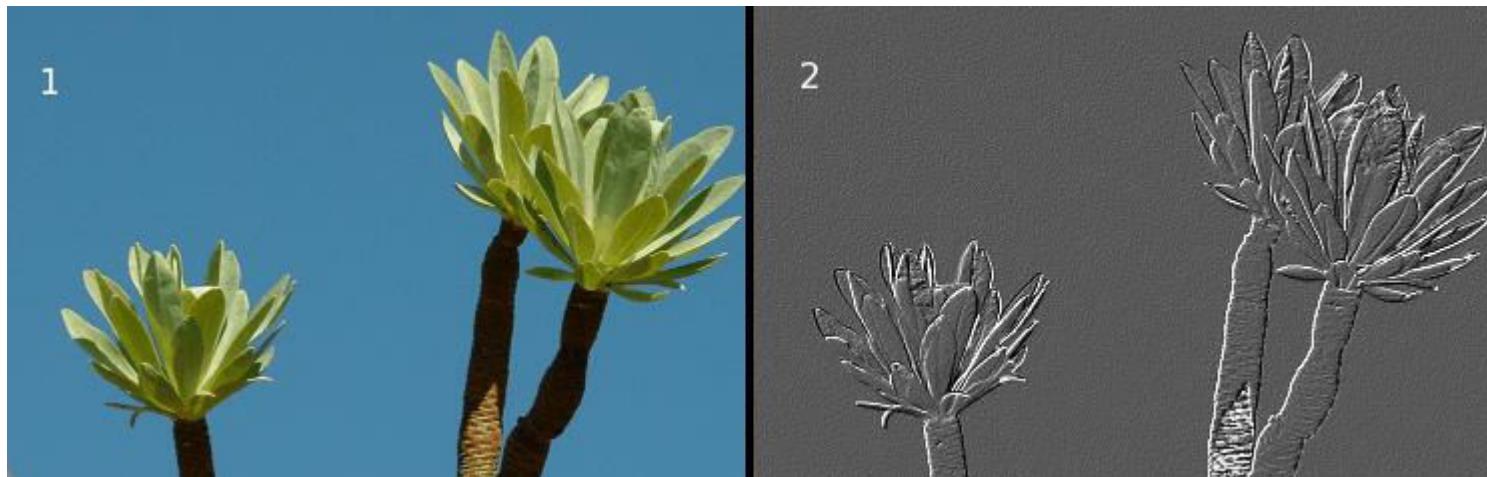
*acumulator = accumulator + $v(x+i,y+j)*k(i,j)$*

$v'(x,y) = accumulator$ (trunchiat la 0..255)

Observații:

- se aplică pe fiecare canal de culoare în parte
- tratare specială pentru pixelii din margine

Filtre de conoluție – Exemplu emboss



$$\begin{pmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{pmatrix} + 127$$

Filtre de conoluție – Alte exemple

Gaussian Blur

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}_{/16}$$

Sharpen

$$\begin{pmatrix} 0 & -2 & 0 \\ -2 & 11 & -2 \\ 0 & -2 & 0 \end{pmatrix}_{/3}$$

Edge Detection

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}_{+127}$$

Formate de stocare

BMP (Microsoft Windows Bitmap)

- Formatul standard de stocare pe platforma Microsoft Windows
- Suportă date necomprimate sau comprimate folosind algoritmul RLE
- Monocromă sau în culori pe 4, 8, 16, 24 sau 32 de biți
- Suportă palete de culori

JPEG (Joint Photographic Experts Group)

- Stocare comprimată cu pierdere de informație conform standardului JPEG
- Rate de compresie diferite selectable de către utilizator
- Utilizat pentru imagini fotografice (cu gradații fine de culoare)
- Nu este potrivit pentru
 - text, linii sau alte imagini care prezintă un contrast foarte mare
 - Editări multiple (se pierde calitate la fiecare etapă de compresie / decompresie)

Formate de stocare

GIF (Graphics Interchange Format)

- Folosit în special pentru transferul imaginilor de maxim maxim 64K x 64K
- Pretabil pentru diagrame, text logo-uri (contrast puternic și număr limitat de culori)
- Suportă maxim 256 culori prin intermediul unei palete de culori
- Poate stoca mai multe cadre (pentru animație)
- Algoritm de compresie fără pierdere de informație Lempel-Ziv-Welch (LZW)

TIFF (Tag Image File Format)

- Format portabil și extensibil utilizat în special pentru imagini scanate
- Suportă stocarea mai multor imagini într-un singur fișier
- Suportă mai mulți algoritmi de compresie (RLE, LZW sau JPEG)

Alte formate: ICO - Icon Resource File, PNG - Portable Network Graphics, DIB - Device Independent Bitmap, PCX - PC PaintBrush File Format

Compresia Run-length encoding (RLE)

Sevențele de valori identice consecutive sunt înlocuite cu perechi de forma
(valoare, număr apariții)

Exemplu:

AAAAAAAAAAAAAAABBBBBBBBBAAAAAAAAAAABB BBBBCCC

14A10B13A6B3C

Caracteristici

- Rată mică de compresie
- Se pretează pentru imagini cu zone mari de aceeași culoare
- Utilizat în special pentru fișiere BMP cu paletă de culori

Compresia Lempel–Ziv–Welch (LZW)

Algoritm de compresie universal bazat pe dicționar

Descriere compresie:

1. Se construiește dicționarul inițial (toate sirurile de lungime 1)
2. Se caută cel mai lung sir W din dicționar care se potrivește cu sirul de la intrare
3. Se elimină W din sirul de intrare
4. Se adaugă W urmat de următorul caracter în dicționar
5. Se continuă cu pasul 2

Decompresie: se parurge sirul codificat și se reconstruiește dinamic dicționarul

Variante: coduri de lungime variabilă, cod pentru reinițializare dicționar

Utilizat pentru fișiere de tip GIF, PNG

Compresia Huffman

Algoritm universal de compresie

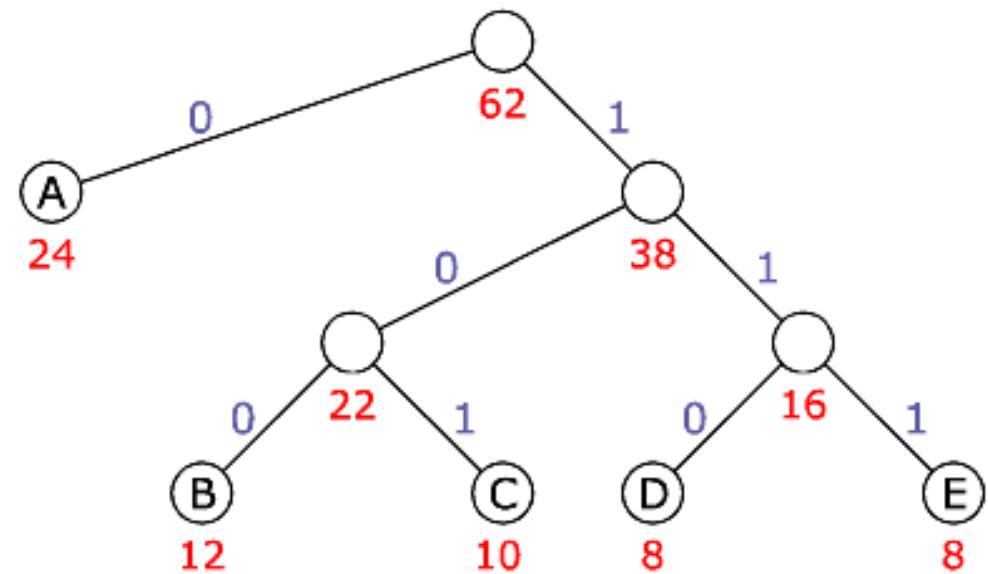
Codificare optimă de lungime variabilă pentru fiecare simbol în funcție de frecvența de apariție

Datele salvate:

- dicționarul
- datele originale recodificate

Decodificare:

- translație simbol cu simbol pe baza dicționarului salvat



Compresia JPEG

Compresie specializată cu pierdere de informație pentru imagine raster

Rezultate foarte bune pentru fotografii (variații fine de luminozitate și culoare)

Tipuri de compresie:

- **secvențial** – codaj bazat pe transformarea cosinus discretă cu blocurile procesate în ordinea apariției
- **progresiv** – codaj bazat pe transformarea cosinus discretă cu blocurile procesate prin mai multe treceri asupra imaginii
- **progresiv fără pierdere** – folosește doar algoritmi de compresie fără pierdere de informație
- **progresiv ierarhic** – codifică imaginea la rezoluții din ce în ce mai mari

Etapele compresiei JPEG

Etapele de **compresie** JPEG File Interchange Format (JFIF)

1. Translatarea modului de culoare din RGB în $Y'C_BC_R$
2. Reducerea rezoluției pentru componente C_B și C_R
3. Imaginea se descompune în blocuri de dimensiune 8x8 pixeli
4. Se aplică transformata cosinus discretă pe fiecare bloc în parte
5. Aplicarea matricei de cuantizare (pierdere de informație)
6. Blocurile rezultate în urma cuantizării sunt comprimate folosind RLE și Huffman

Decodificare: se aplică pașii în ordine inversă

JPEG -Transformata cosinus discretă

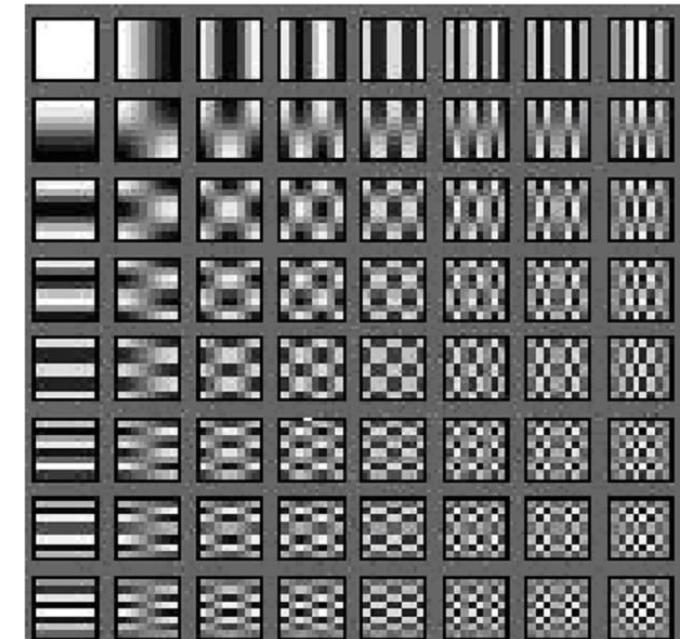
$$\text{DCT: } G_{ij} = \frac{1}{4} C_i C_j \sum_{x=0}^7 \sum_{y=0}^7 p_{xy} \cos\left(\frac{(2y+1)j\pi}{16}\right) \cos\left(\frac{(2x+1)i\pi}{16}\right)$$

$$\text{IDCT: } p_{xy} = \frac{1}{4} \sum_{i=0}^7 \sum_{j=0}^7 C_i C_j G_{ij} \cos\left(\frac{(2y+1)j\pi}{16}\right) \cos\left(\frac{(2x+1)i\pi}{16}\right)$$

unde

$$C_i = C_j = \frac{1}{\sqrt{2}}, \quad \text{dacă } i = j = 0$$

$$C_i = C_j = 1, \quad \text{în rest}$$



JPEG -Transformata cosinus discretă

139	144	149	153	155	155	155	155		235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
144	151	153	156	159	156	156	156		-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
150	155	160	163	158	156	156	156		-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
159	161	162	160	160	159	159	159		-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
159	160	161	162	162	155	155	155		-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
161	161	161	161	160	157	157	157		1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
162	162	161	163	162	157	157	157		-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
162	162	161	161	163	158	158	158		-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

JPEG - Cuantizare

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

$$Cq(u,v) = \text{round}[C(u,v)/N(u,v)]$$

$$\begin{aligned} 235.6/16 &\rightarrow 15 \\ -22.6/12 &\rightarrow -2 \end{aligned}$$

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Exemplu: <https://squoosh.app/>

Canvas – Salvare Imagine

Se utilizează funcția:

canvasElement.toDataURL(type, encoderOptions)

- *type*: formatul imaginii (**image/png**, image/jpeg, ...)
- *encoderOptions*: opțiuni dependente de tipul imaginii și browser (exemplu: un număr între 0 și 1 reprezentând calitatea pentru jpeg)

Șirul de caractere rezultat poate fi utilizat ca URL pentru elemente de tip IMG sau A:

```
var img = document.createElement('img');
img.src = canvas.toDataURL('image/jpeg', 0.4)
document.body.append(img);
```

```
let a = document.createElement('a');
a.download = 'imagine.jpg';
a.href = canvas.toDataURL('image/jpeg', 0.1);
a.click();
```

3. Animăție

Modificarea rapidă a imaginii vizualizate prin modificarea poziției, formei sau dimensiunii unui obiect din imagine

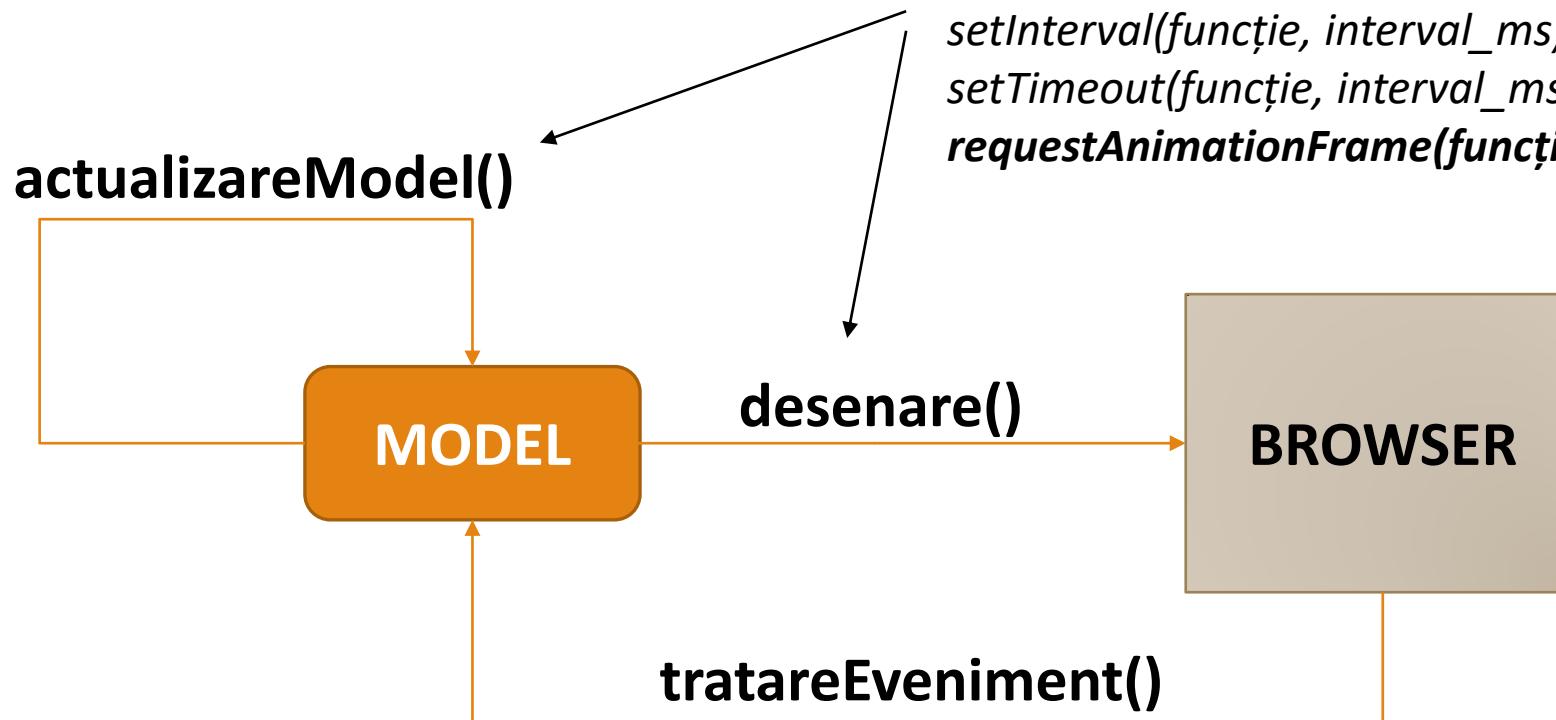
Stocarea numerică a animației presupune reținerea elementelor independente ce compun mișcarea în raport cu factorul timp.

Crearea iluziei de mișcare se realizează prin afișarea rapidă de imagini statice ușor modificate

Tehnici principale:

- Tehnica filmului
- Cadre cheie
- Schimbarea culorii

Animație - JavaScript



4. Grafică vectorială

Bazată pe descrierea matematică a obiectelor componente ale imaginii

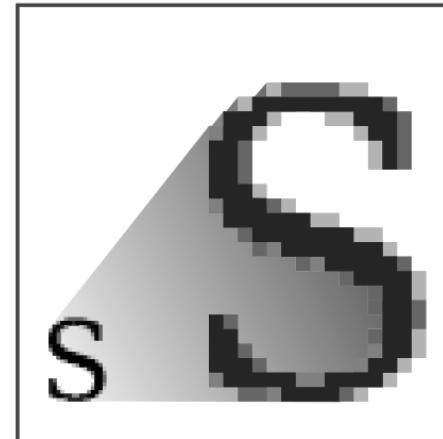
Avantaje:

- Menține semantica – editare la nivel de obiect grafic
- Dimensiune redusă
- Independente de scara de vizualizare

Dezavantaj

- Nu poate reprezenta fidel orice fel de informație

GIMP



BITMAP
.jpeg .gif .png

INKSCAPE



OUTLINE
.svg

Formate de reprezentare și stocare

SVG (Scalable Vector Graphics)

- Format generic bazat pe XML pentru reprezentări vectoriale 2D
- Suportă animație și interactivitate

DXF (Drawing Exchange Format)

- formatul vectorial lansat de firma Autodesk pentru produsul software AutoCAD

EPS (Encapsulated Post Script)

- formatul firmei Adobe pentru imagini vectoriale
- se bazează pe un limbaj de descriere numit Post Script

SHP (Shapefile)

- formatul firmei ESRI pentru descrierea datelor spațiale de tip: punct, polilinie și poligon
- utilizat la reprezentarea elementelor geografice în sisteme de tip GIS

SVG – Scalable Vector Graphics

Poate fi:

- inclus direct într-o pagină HTML
- controlat prin intermediul CSS și JavaScript

Exemplu:

```
<!DOCTYPE html>
<html>
<body>
    <svg width="400" height="180">
        <rect x="50" y="20" width="150" height="150"
style="fill:red;stroke:black;stroke-width:5;opacity:0.5"/>
    </svg>
</body>
```

Sistem de coordonate implicit – similar *canvas*

SVG – Elemente de bază

Linii

```
<line x1="start-x" y1="start-y" x2="end-x" y2="end-y">
```

Dreptunghiuri

```
<rect x="start-x" y="start-y" width="width" height="height"/>
```

Cercuri

```
<circle cx="center-x" cy="center-y" r="radius"/>
```

SVG – Elemente de bază

Elipse

```
<ellipse cx="center-x" cy="center-y" rx="radius-x" ry="radius-y"/>
```

Poligoane

```
<polygon points="x1,y1 x2,y2 ..."/>
```

Poli-linii

```
<polyline points="x1,y1 x2,y2 ..."/>
```

Text

```
<text x="start-x" y="start-y">conținut</text>
```

SVG - Atribute

Setate prin intermediul CSS

Atribute de bază

- stroke – culoare linie
- stroke-opacity – opacitate linie
- stroke-width – dimensiune linie

- fill – culoare suprafață
- fill-opacity – opacitate suprafață

SVG – Grupare și reutilizare

Grupare elemente

```
<g id="id_grup">... <!-- elemente --> </g>
```

Definire elemente fără afișare

```
<defs>... <!-- definire grupuri --> </defs>
```

Reutilizare

```
<use xlink:href="#id_grup" x="30" y="14"/>
```

SVG – Transformări

Se aplică pentru un element prin intermediul atributului ***transform***

Modifică întreg sistemul de coordinate

Transformări disponibile:

- scale(sx[, sy])
- translate(tx, ty)
- rotate(unghi[, cx, cy])
- skewX(unghi) / skewY(unghi)

SVG - Efecte

Se aplică pentru un element prin intermediul atributului ***transform***

Sintaxa:

```
<filter id="F">
  <anyParticularPrimitive1>
  <anyParticularPrimitive2>
  ...
  <anyParticularPrimitiveN>
</filter>
<anyParticularSVGObjectOrGroup filter="url(#F)"/>
```

Exemplu:

```
<filter id="filtru">
  <feConvolveMatrix kernelMatrix="3 3 3      0 0 0      -3 -3 -3" />
  <feGaussianBlur stdDeviation="3" />
</filter>
<circle cx="10" cy="20" r="17" filter="url(#filtru)"></use>
```

Lista filtre: https://developer.mozilla.org/en-US/docs/Web/SVG/Element#Filter_primitive_elements

SVG – Manipulare din JavaScript

Similar cu manipularea elementelor HTML

Particularități

- la construirea elementului trebuie specificat namespace-ul pentru SVG:
 - `document.createElementNS("http://www.w3.org/2000/svg", „TAG_SVG")`
- în jQuery se utilizează funcția `attr` pentru modificarea atributelor (în loc de `.width`, `.height`, ...)
- Folosind JavaScript simplu:
 - `elem.setAttributeNS(null, 'nume', 'valoare');`
 - `val = elem.getAttributeNS(null, 'nume');`

Exemplu:

```
$(document.createElementNS("http://www.w3.org/2000/svg", "rect"))
```

```
.attr({x:160, y:160, width:12, height:12})
```

```
.appendTo($("#desen"));
```

III. Sunet

1. Notiuni generale
2. Numerizarea sunetului
3. Formate audio
4. Compresia sunetului
5. Sunetul în context Web

1. Noțiuni generale

Sunetul este o vibrație propagată printr-un mediu material sub forma unei unde mecanice.

- Din punct de vedere fiziologic: senzația produsă asupra organului auditiv de către vibrațiile materiale ale corpurilor și transmise pe calea undelor acustice;
- Urechea umană percepse vibrații în intervalul 20-20000Hz
- Zgomot: caz particular de sunet caracterizat prin lipsa încărcăturii informaționale

Redare / receptare sunet

- Prin intermediul difuzorului și microfonului
- Ambele fac conversie semnal electric – vibrație mecanică
- Folosesc principiile inducției electomagnetice

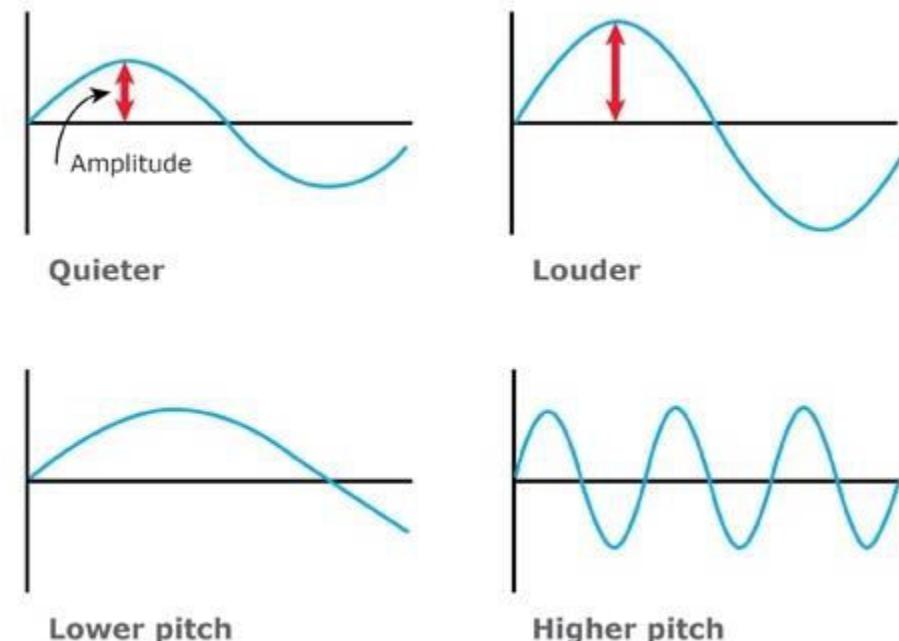
Reprezentare și caracteristici

Reprezentare:

- Axa X: timp
- Axa Y: presiune (0 – presiunea aerului în repaus)

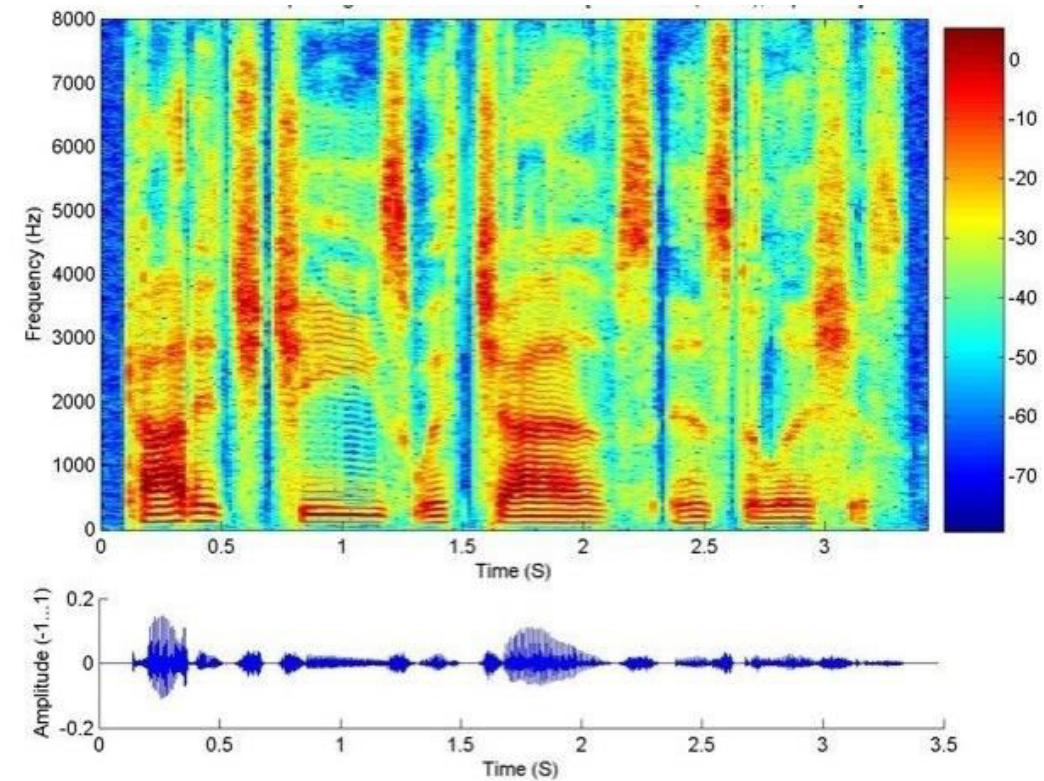
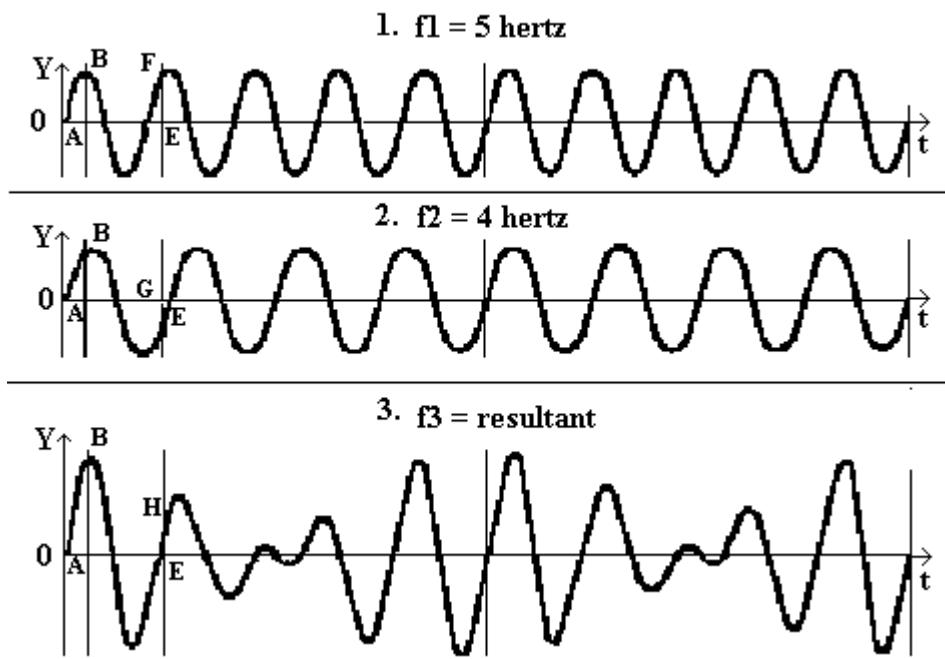
Amplitudine: măsoară dimensiunea vibrației / volumul sunetului

Frecvență: măsoară viteza vibrației / tonul sunetului



Compunere / descompunere

26.8.4.4



2. Numerizarea sunetului

Presupune stocarea și prelucrarea sunetului în format digital

Etape:

- Convertirea sunetului în semnal electric
- Eșantionarea și cuantificarea semnalului
- Stocarea informației numerice pe un suport de memorie externă conform unui format

Avantaje

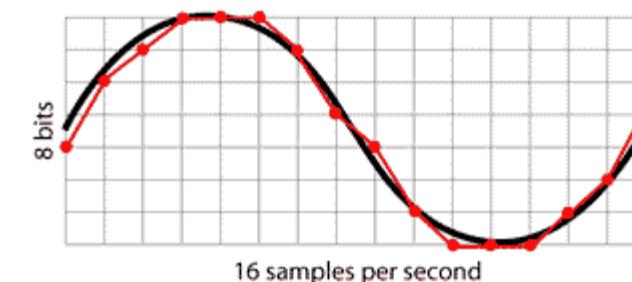
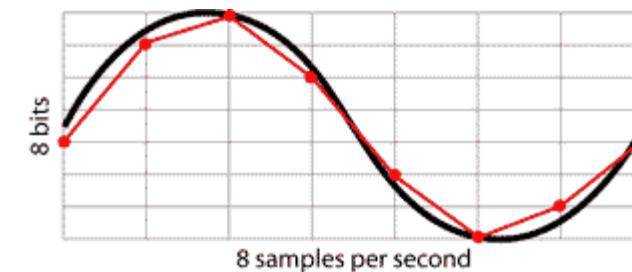
- Stocare mai ușoară
- Permite analiza și procesarea numerică a sunetului
- Nu se degradează în timp sau la copieri repetate

Eșantionare

Prin eșantionare se înțelege procesul de segmentare, cu o perioadă fixă, a semnalului audio analog.

Frecvența de eșantionare – rezoluția orizontală

- Se determină pe baza teoremei lui Nyquist (minim dublul frecvenței maxime a sunetului)
- Rate de eșantionare uzuale:
 - 8 kHz – semnal telefonic
 - 11 kHz – radio AM
 - 22 kHz – radio FM
 - 44 kHz – audio CD



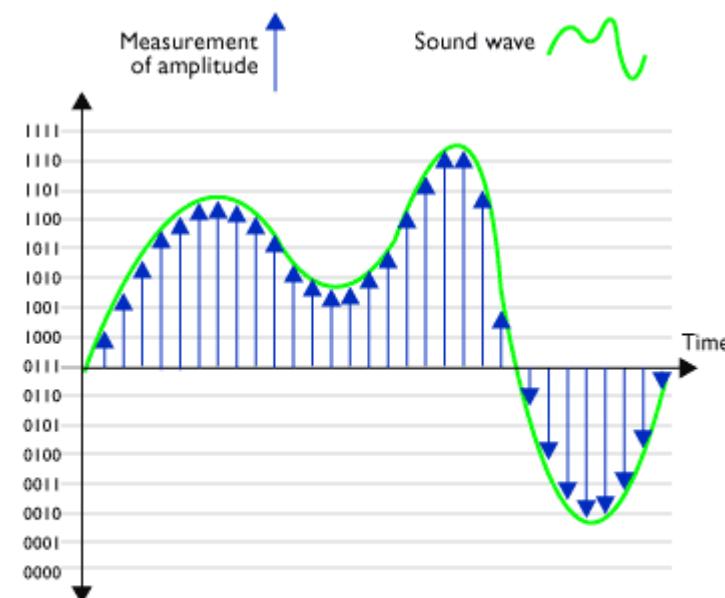
Cuantificare

Cuantificarea presupune asocierea unei valori numerice corespunzătoare amplitudinii semnalului pentru fiecare interval de timp.

Calitatea este influențată de numărul de biți alocați pentru fiecare eşantion (uzual 8 sau 16 biți pentru stocare și 16 – 32 pentru procesare)

Redarea sunetului digital:

- Se reconstruiește sinusoida originală prin interpolarea valorilor numerice stocate
- Prin intermediul unui convertor digital - analog



Each measurement is assigned a number (byte) according to its amplitude. The end result is a file comprising a string of bytes, eg ...
1001 1110 0001 1010 0111 0100 1111 1101 etc

3. Formate audio

WAVE – formatul standard de fișier audio pentru Microsoft și IBM; conține sunet în reprezentare PCM necomprimat;

AIFF (Audio Interchange File Format) – formatul standard pentru audio digital utilizat pe platformele Apple (variante: necomprimat / comprimat);

MPEG (Moving Picture Experts Group) Audio - format standard pentru sunetul digital comprimat; parte a standardului MPEG de codificare a semnalului audio-video; cea mai cunoscută variantă a lui este MP3

OGG/VORBIS – algoritm de compresie perceptuală, similar MP3, dar cu implementare open source; folosește formatul de fișier OGG

4. Compresia sunetului

Cel mai utilizat algoritm de compresie: **MPEG-1 sau 2 Audio Layer III (MP3)**

Folosește codificare perceptuală

- Elimină din rezultat sunetele care nu pot fi percepute de către urechea umană

Sunetele imperceptibile sunt eliminate pe baza unui model psihooacustic care exploatează fenomenele de:

- Mascare a frecvențelor
- Mascare temporală

Compresia sunetului - mascarea

Mascarea frecvențelor

- Sunt eliminate sunetele cu frecvența mai mare de 16-18 KHz
- Sunt eliminate sunetele de intensitate scăzută, care apar concomitent cu sunete de intensitate înaltă, dacă sunt în benzi de frecvență alăturate (cele cu intensitate scăzută sunt măcate de cele cu intensitate înaltă)

Mascarea temporală

- Se elimină sunetele de intensitate mică care urmează după sunete de intensitate puternică în cadrul unui interval de timp
- Sunetele de intensitate mică nu pot fi percepute după sunete de intensitate puternică datorită inerției timpanului

Compresia MP3 - Etape

1. Utilizarea de filtre pentru separarea sunetului în 32 sub-benzi de frecvență
2. Transformări
 - FFT: se aplică modelul psihoaesthetic pentru determinarea factorului de scalare
 - DCT: proces de cuantizare cu factorul de scalare determinat anterior
3. Codificare Huffman pentru valorile cuantizate
4. Componerea fluxului final de biți

5. Sunetul în context Web

La nivel de HTML – tag `<audio>`

Exemplu:

```
<audio controls="controls">
    <source src="test.wav" type="audio/wav">
    <source src="test.mp3" type="audio/mpeg">
</audio>
```

Formate suportate:

- mp3 – type = audio/mpeg
- wav – type = audio/wav
- ogg – type = audio/ogg

Audio - atributे

Elementul <audio>:

autoplay (bool) – redarea automată a sunetului

controls (string) – controalele de redare sunt afișate dacă atributul este prezent

loop (bool) – permite redarea continuă a sunetului

src (string) – permite specificarea sursei fără utilizarea de tag-uri de tip *source*

Elementul <source>

src (string) – adresa (URL) fișierului audio

type (string) – tipul MIME pentru fișierul audio

Audio – obiectul *HTMLMediaElement*

Proprietăți:

currentSrc – URL-ul absolut al fișierului redat

currentTime – poziția (în secunde) în cadrul fișierului (poate fi modificată)

duration – durata totală a fișierului audio (în secunde)

ended – boolean setat pe *true* la terminarea redării

error – ultima eroare (obiect *MediaError*) sau *null* dacă nu a apărut nici o eroare

paused – boolean setat pe *false* la oprirea redării

readyState – indică starea curentă a elementului

volume – permite citirea / modificarea volumului

Audio – obiectul *HTMLMediaElement*

Metode:

canPlayType(type) – permite aplicației să determine dacă browser-ul curent suportă un anumit tip de fișier audio

load() – pornește procesul de descărcare a fișierului audio de pe server; este obligatoriu să fie apelat înainte de începerea redării folosind metoda *play()*

pause() – oprește redarea (cu păstrarea poziției curente)

play() – pornește redarea de la poziția curentă

Audio – obiectul *HTMLMediaElement*

Evenimente:

canplay – a fost încărcată o parte din fișier și poate fi pornită redarea

ended – redarea s-a terminat

pause – redarea a fost oprită

play – redarea a început

volumechange – modificare de volum

waiting – operația curentă este suspendată pentru a încărca date de pe server

Web Audio API

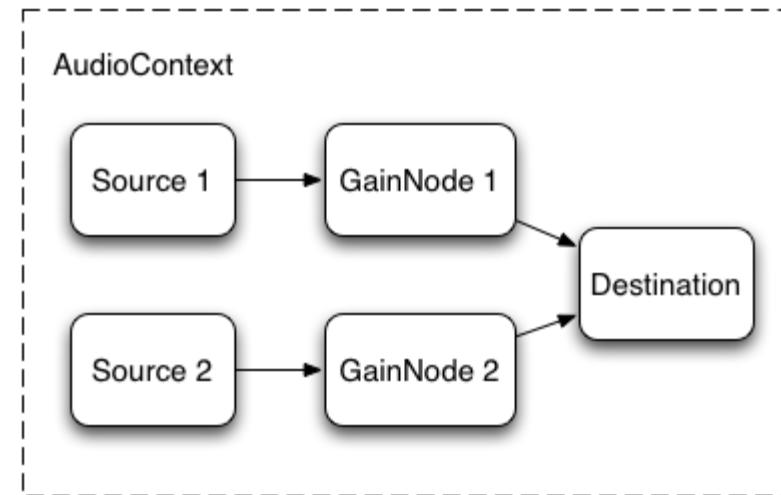
Permite generarea și procesarea de sunet în cadrul unui browser web.

Operațiile:

- se desfășoară în interiorul unui **context audio**
- sunt specificate prin intermediul unui **graf de rutare**

Tipuri de noduri:

- Sursă: preiau datele audio dintr-un tag audio sau generează sunete
- Procesare: prelucrează datele primite la intrare
- Destinație: nodul final al graficului



Web Audio API

AudioContext – reprezintă un graf de procesare audio

- Proprietăți: `.currentTime` (poziția curentă în secunde), `.destination` (referința la nodul destinație)
- Metode:
 - `createMediaElementSource(audioElement)`: construiește un nod sursă pe baza unui element `<audio>`
 - `createOscillator()`: construiește un nod sursă pentru generarea de sunete
 - `createGain()`: construiește un nod de procesare pentru ajustarea volumului
 - `createAnalyser()`: permite analiza sunetului (descompunere Fourier)
 - `createScriptProcessor(bufferSize)`: construiește un nod pentru procesare JavaScript
 - `.close()`:

AudioNode – reprezintă un nod din cadrul grafului (din care sunt moștenite celelalte noduri)

- `.connect(node)`: conectează ieșirea nodului curent la intrarea nodului primit ca parametru
- Diverse metode și proprietăți în funcție de tipul nodului

IV. Video

1. Noțiuni generale
2. Compresia video
3. Video în context web

1. Noțiuni generale

Video digital – cuprinde totalitatea tehniciilor de captură, procesare și stocare a imaginilor în mișcare (precum și a sunetului asociat) prin intermediul unui dispozitiv de calcul.

Avantaje video digital:

- Poate fi procesat prin intermediul calculatorului
- Păstrare în timp și rezistență la copieri repetitive
- Poate fi transmis la distanță

Caracteristici video

Rezoluția

Spațiul de culoare și numărul de biți per pixel

Numărul de cadre pe secundă

Modul de afișare (întrețesut sau progresiv)

Calitatea compresiei

Formate video

Container – specifică structura de stocare a componentelor video (imagine + audio) și a datelor asociate (metadate, subtitrări, ...)

- Advanced Systems Format – **ASF**: container dezvoltat de Microsoft care poate conține fluxuri codate cu orice codec (Extensii: .ASF, .WMA, .WMV)
- Audio Video Interleave – **AVI**: container mai vechi dezvoltat de Microsoft pe baza Resource Interchange File Format – RIFF (stochează datele în secțiuni identificate prin markere FourCC)
- **MP4** – MPEG-4 Part 14: dezvoltat de către Motion Pictures Expert Group și utilizat inițial de către QuickTime (video H.264, audio AAC)
- **AVCHD** – format utilizat în special de către camerele video (video H.264 AVC și sunet AC3 sau PCM)
- **Matroska / OGG**: formate deschise; pot conține mai multe fluxuri audio / video

Formate video

Codec – specifică modalitatea de compresie / decompresie pentru un flux video / audio în cadrul unui container

- **H.264 / MPEG-4 AVC** – cel mai popular (utilizat pentru Web, BluRay, camere video)
- **H.262 / MPEG-2** – formatul standard pentru DVD
- **Windows Media Video** – format dezvoltat de către Microsoft
- **MJPEG (Motion JPEG)** – format mai vechi bazat pe compresia JPEG

2. Compresia video

Se bazează pe reducerea redundanței din cadrul fluxului video

Redundanță spațială (intra-cadru)

- tipul de redundanță identificat și eliminat de algoritmii de compresie a imaginilor

Redundanță temporală (inter-cadru)

- redundanță identificată între două cadre consecutive (de exemplu, prin compararea a două cadre se observă că majoritatea pixelilor își păstrează valoarea)

Compresia MPEG

Algoritm de compresie video

- Hibrid
 - Transformata Cosinus Discretă – similar JPEG pentru reducerea redundanței spațiale
 - Codaj Huffman – pentru comprimarea coeficienților TCD
 - Codificarea mișcării – pentru reducerea redundanței temporale
 - Codaj RLE
- Asimetric
 - Timpul de codare este mult mai mare decât cel de decodare

Etapele de compresie

Împărțirea imaginii în blocuri

- 16x16 luminanță
- 8x8 crominanță (culoare)

Compresie pe baza DCT pentru reducere spațială

Aplicarea tehniciilor de compensare a mișcării pentru reducere temporală

Faza finală de codare pe două dimensiuni folosind Run Length Encoding

Tipuri de cadre

<I> Intra-picture/frame/image

- Cadrele cheie
- Necesare pentru căutare și poziționare
- Compresie moderată

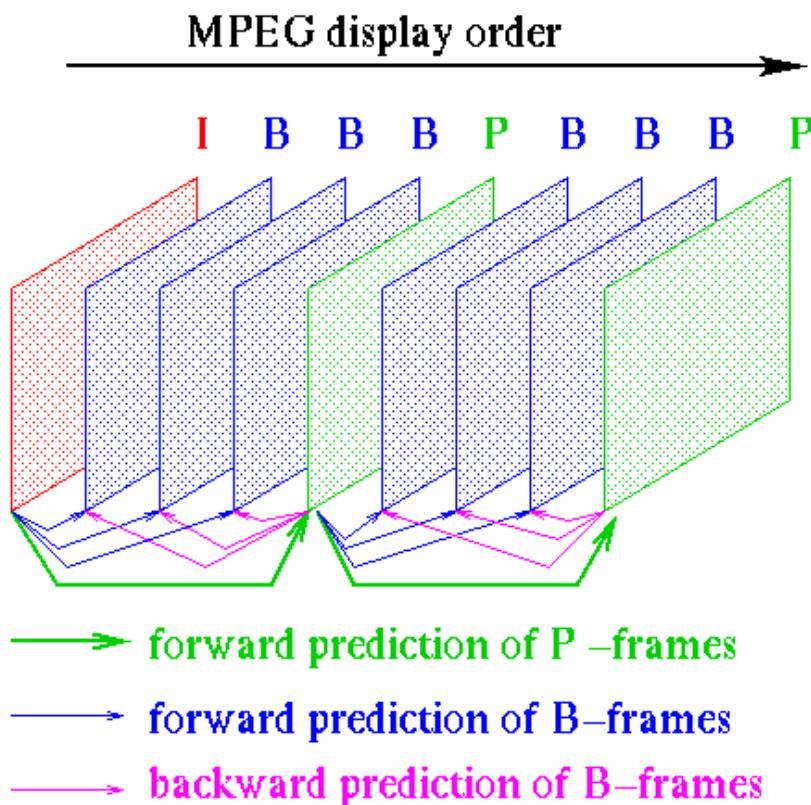
<P> Predicted pictures

- Codate cu referință la un cadru anterior
- Folosite ca referință pentru cadre ulterioare

* Bi-directional prediction (interpolated pictures)*

- Necesită cadre anterioare și viitoare pentru refacere
- Compresie mare

Tipuri de cadre



4. Video în context Web

La nivel de HTML – tag <video>

Exemplu:

```
<video controls="controls">
    <source src="test.webm" type="video/webm">
    <source src="test.mp4" type="video/mp4">
</video>
```

Formate suportate:

- webm – type = video/webm
- mp4 – type = video/mp4
- .ogg – type = video/ogg

Atribute / manipulare din JavaScript

Similar cu elementul audio

- Atribute: *autoplay, controls, src, volume, ...*

Reprezentat la nivel DOM prin obiecte de tip `HTMLMediaElement`

- Proprietăți: `currentSrc`, `currentTime`, `duration`, `ended`, `error`, `paused`, `readyState`, `volume`
- Metode: `canPlayType`, `load`, `pause`, `play`
- Evenimente: `canplay`, `ended`, `pause`, `play`, `volumchange`, `waiting`

Exemplu: Încărcare dinamică video

```
var v = $("<video></video>")
    .attr({
        "controls": "hidden",
        "autoplay": "autoplay",
        "src": "media/movie.mp4"
    })
    .load()
    .appendTo($(".body"));

// modificați sursă
v[0].src = "media/test.mp4";
v[0].load();
v[0].play();
```

Exemple: Procesare cadru - 1

```
var video = $("#video")[0];
var canvas = $("#canvasProcesare")[0];
var context = canvas.getContext("2d");
function procesareCadru() {
    //...
    requestAnimationFrame(procesareCadru);
};
requestAnimationFrame(procesareCadru);
```

Exemplu: Procesare cadru - 2

```
var W = canvas.width = video.clientWidth;  
var H = canvas.height = video.clientHeight;  
  
context.drawImage(video, 0, 0, W, H);  
var imageData = context.getImageData(0, 0, W, H);  
  
for (var y = 0; y < H; y++) {  
    for (var x = 0; x < W; x++) {  
        var i = (y * W * 4) + x * 4;  
        // modificați valori imageData.data[i+...]  
    }  
}  
context.putImageData(imageData, 0, 0);  
// alte operații de desenare pe canvas
```

Exemple: Playlist simplu

```
$(function () {
    var lista = ["movie.mp4", "v2.mp4"];
    var index = 0;

    var video = $("#myVideo");

    video.on("ended", function () {
        index = index + 1;
        if (index >= lista.length) {
            index = 0;
        }
        video[0].src = lista[index];
        video[0].load();
        video[0].play();
    });
});
```



Department of Economic Informatics and Cybernetics
Bucharest University of Economic Studies

Multimedia

Liviu-Adrian Cotfas, PhD.



liviu.cotfas@ase.ro

Few words about me...



<https://ro.linkedin.com/in/cotfasliviu>

Optional

Aspecte administrative

Further Reading / Watching

- Cursuri Microsoft Virtual Academy - mva.microsoft.com
 - Free
- Cursuri PluralSight - www.pluralsight.com
 - Free trial
 - Free access (limited period) through Microsoft DreamSpark

5

- slide-urile marcate în laterală cu "Optional" conțin informații adiționale, care nu vor fi solicitate la evaluare (lucrare, proiect și colocviu);
- în acest context, unele dintre slide-urile marcate în laterală cu "Optional" pot fi redactate în limba engleză;
- slide-urile în limba engleză, nu vor fi solicitate la evaluare (lucrare, proiect și colocviu);

Recommended Reading / Watching

- Prezentare curs, Exemple seminar:
 - <https://github.com/liviucotfas/ase-multimedia>
 - <http://online.ase.ro>

Aspecte administrative

Further Reading / Watching

- Cursuri Microsoft Virtual Academy - mva.microsoft.com
 - Free
- Cursuri PluralSight - www.pluralsight.com
 - Free trial
 - Free access (limited period) through Microsoft DreamSpark

Definiție, Caracteristici, Concepte

Definiție

- **Multimedia** este din punct de vedere informatic :
 - orice combinație de **text, imagine, video, sunet și animație**
 - accesibilă utilizatorului prin intermediul sistemului de calcul.
- **Dezvoltarea multimedia** a devenit posibilă ca urmare a Revoluției Digitale, prin intermediul :
 - conversiei analog-digital;
 - compresiei datelor.

Factori declanșatori

- **Revoluția digitală** reprezintă schimbarea de la tehnologia mecanică și electronică analogică la electronica digitală care a început la sfârșitul anilor 1950 până la sfârșitul anilor 1970, odată cu adoptarea și proliferarea computerelor digitale și păstrarea înregistrărilor în format digital.

Conversia analog - digital

- Resursele utilizate trebuie convertite în format digital pentru:
 - stocare
 - procesare
 - includere în aplicații
- Resursele sunt convertite în format analog la redare

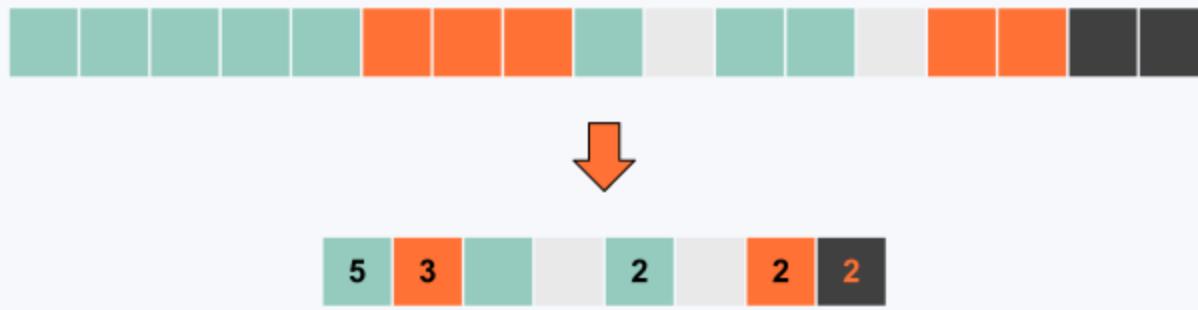
Compresia

- Există două categorii majore de algoritmi de compresie :
 - Compresie lossless (fără pierderi)
 - Compresie lossy (cu pierderi)
- Algoritmii de compresie utilizați în multimedia sunt de obicei **asimetrici** – timpul de compresie este mai mare decât timpul de decompresie.

Compresia lossless

- utilizează o codificare mai eficientă pentru a reduce dimensiunea fișierului, păstrând în același timp toate datele originale. Când fișierul este decompresionat, acesta va fi identic cu originalul.
- Run Length Encoding - RLE
 - una dintre strategiile mai simple pentru a realiza compresia lossless
 - poate fi folosit pentru a comprima fișiere de imagine bitmap(ex: format *pcx). Imaginele bitmap pot deveni cu ușurință foarte mari, deoarece fiecare pixel este reprezentat cu o serie de biți care oferă informații despre culoarea sa. RLE generează un cod pentru a „semnaliza” începutul unei linii de pixeli de aceeași culoare. Informațiile despre culoare sunt apoi înregistrate o singură dată pentru fiecare pixel. De fapt, RLE spune computerului să repete o culoare pentru un anumit număr de pixeli adiacenți, mai degrabă decât să repete aceleași informații pentru fiecare pixel de mai multe ori. Fișierul comprimat RLE va fi mai mic, dar va păstra toate datele originale ale imaginii – este "lossless".

Run Length Encoding - RLE



Compresia Lossy

- numărul de biți din fișierul original este redus și unele date se pierd. Compresia Lossy nu este o opțiune pentru fișierele constând din text și numere, aşa-numitele informații *alphanumerice*. În cazul acestora pierderea unei singure litere sau număr ar putea modifica cu ușurință semnificația datelor
- este adesea posibil să se mențină imagini sau sunete de înaltă calitate cu mai puține date decât era prezent inițial (util în special pentru multimedia)
- exploatează limitele perceptiei umane

Compresia Lossy

- Example:
 - JPEG – imagini;
 - MPEG – sunet și video.
- compresie MP3:
 - analizează fișierul audio și elimină date care nu sunt critice pentru o redare de înaltă calitate;
 - elimină frecvențele care nu pot fi percepute de auzul uman. De asemenea, poate evalua două sunete care rulează în același timp și poate elimina sunetul mai puțin perceptibil. Aceste tipuri de date pot fi eliminate fără impact semnificativ asupra calității;
 - poate reduce cu un factor de 10 cantitatea de date necesară pentru a reprezenta înregistrările audio digitale, la o calitate similară cu cea a sunetului original necomprimat pentru majoritatea ascultătorilor.

Aplicații Multimedia

- Caracteristici:
- componentele sunt accesibile prin intermediu unui sistem de calcul;
- datele utilizate sunt în format digital (NU analogic);
- elementele sistemului sunt integrate într-o interfață unitară;
- grad ridicat de interacțiune cu utilizatorul.

Abordări pentru construirea de aplicații multimedia

- **Multimedia authoring – high-level**
 - includ componente preprogramate ce permit recunoașterea mai multor formate de resurse multimedia, instrumente pentru generarea animațiilor, pentru implementarea conceptelor de hypertext, hypermedia;
 - accentul cade pe scenariul de derulare a aplicației și pe sincronizarea resurselor;
 - bazat pe concepte precum axa timpului / carte / diagrame de flux;
 - Exemple: Flash, PowerPoint.
- **Multimedia programming – low-level**
 - accentul se pune pe procesarea directă a resurselor prin intermediul unui limbaj de programare precum C, C#, JavaScript sau Java și a unor biblioteci specializate.

Classification of multimedia applications

- Mai multe criterii de clasificare bazate pe :
 - domeniu: economie, educație, publicitate, medicină, industrie, divertisment, sisteme de navigație și informații, comunicări;
 - interactivitate: interactiv, static;
 - localizarea componentelor: local, la distanță (video-streaming, distant learning).

Precondiții hardware / software

Precondiții hardware

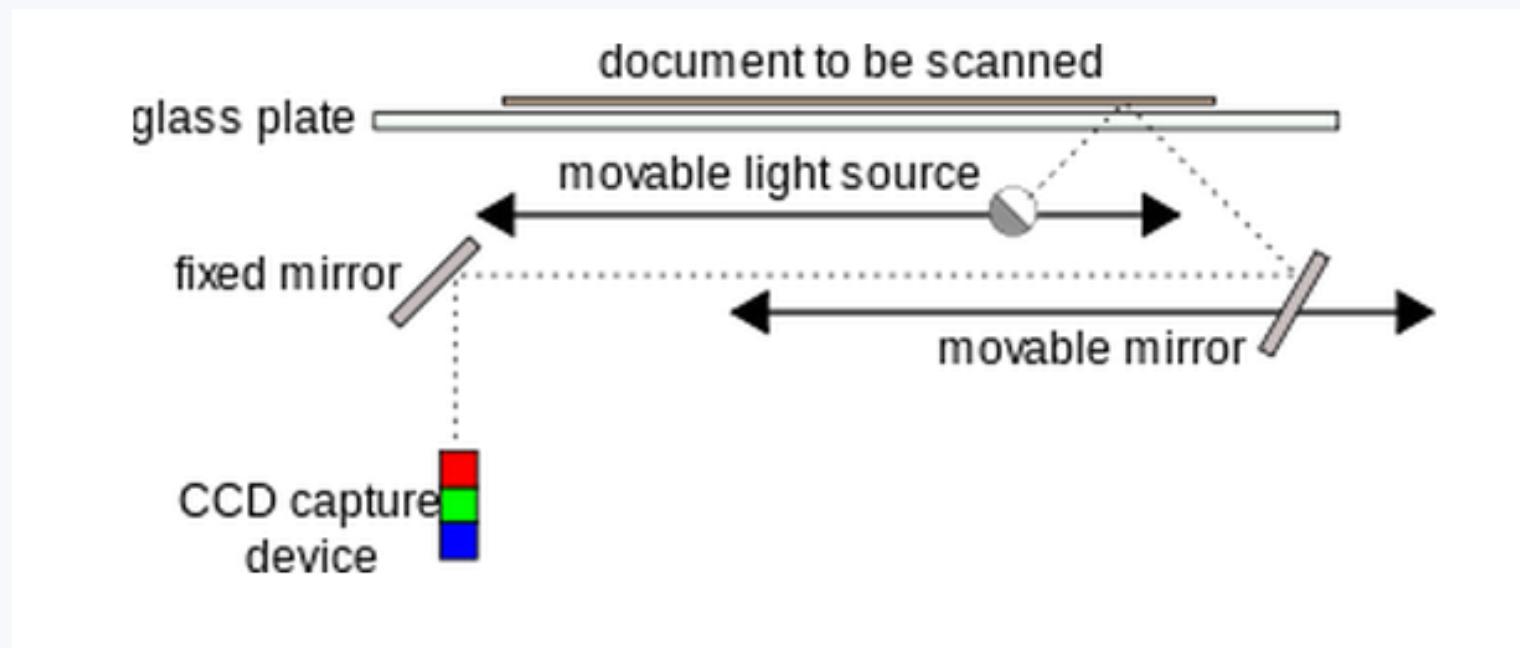
- Echipamente hardware pentru achiziție / procesare:
 - Imagine;
 - Sunet;
 - Video.

Dispozitive periferice - Imagine

- Scanner
 - transformă informația luminoasă în informație electrică, iar ulterior aceasta este convertită și salvată sub formă digitală.
 - tipuri: flatbed / rotativ
 - operație: o lumină trece peste text sau imagine, iar lumina se reflectă înapoi pe un senzor **CCD** (charge-coupled device). Un senzor CCD este un dispozitiv electronic care captează semnauul analogic. Semnalul analogic este apoi convertit într-un semnal digital de către un alt dispozitiv numit **ADC** (analog-to-digital converter) și transferat prin intermediul unei interfețe (în general USB) către RAM.
 - Recunoașterea optică a caracterelor (OCR) este procesul de conversie a textului tipărit într-un fișier digital care poate fi editat într-un procesor de text.

Dispozitive periferice - Imagine

- scanner

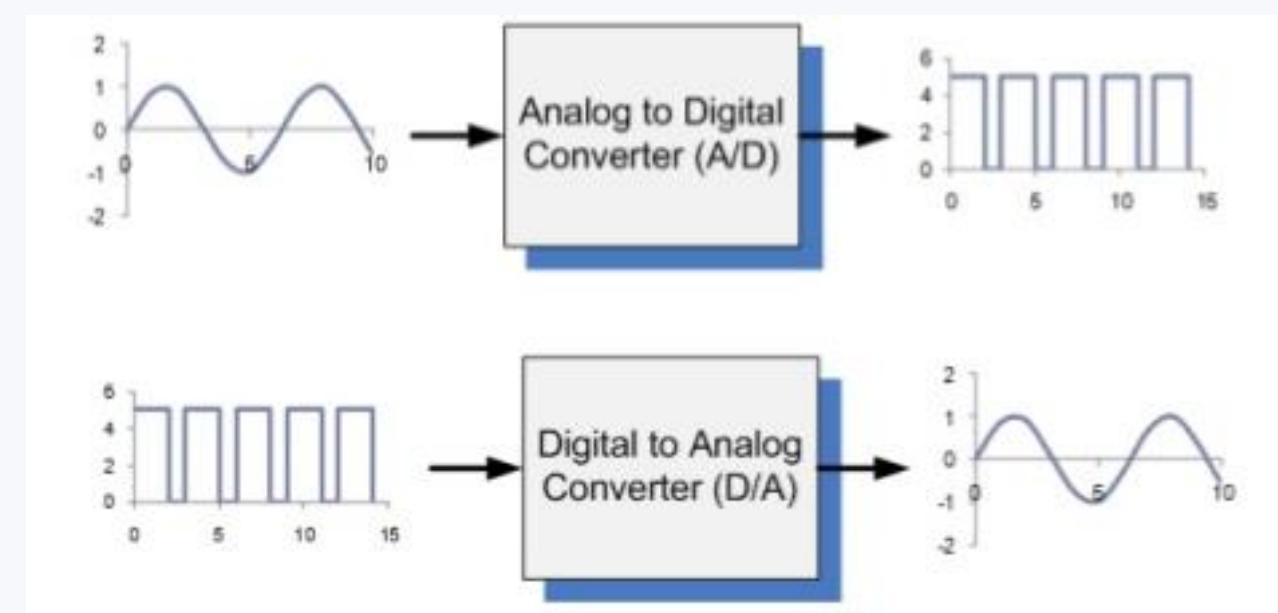
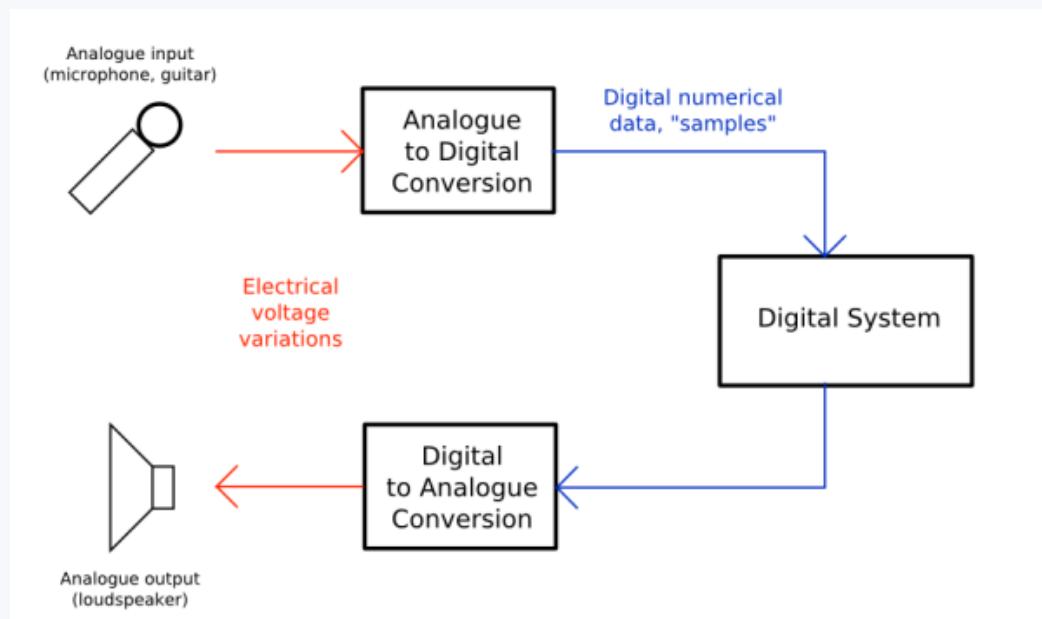


Dispozitive periferice - Imagine

- Camere foto
 - Când declanșatorul camerei este deschis pentru a captura o imagine, lumina trece prin obiectivul camerei. Imaginea este focalizată pe un CCD, care generează un semnal analogic.
 - Semnalul analogic este convertit în formă digitală de un ADC și apoi trimis la un procesor de semnal digital (DSP) chip care ajustează calitatea imaginii și o stochează în memoria încorporată a camerei sau pe un card de memorie.

Dispozitive periferice - Sunet

- Placă de sunet
 - convertește semnalul analogic de la microfon la o reprezentare digitală;
 - convertește reprezentarea digitală într-o reprezentare analogică care poate fi redată de difuzoare.



Dispozitive periferice - Video

- Placă video
- Cameră video digitală

Precondiții software

- **Drive** - program de computer care operează sau controlează un anumit tip de dispozitiv care este atașat la un computer. Un driver oferă o interfață software pentru dispozitivele hardware, permitând sistemelor de operare și alte programe de computer să acceseze funcțiile hardware fără a fi nevoie să știe detalii precise despre hardware-ul utilizat.
- **Extensii ale sistemului de operare:**
 - biblioteci specializate pentru controlul resurselor multimedia;
 - instrumente de bază pentru redarea conținutului (ex: Groove Music).
- **Software specializat**

Software specializat

- Există două tipuri majore de software pentru dezvoltare multimedia:
 - **Media-specific** applications sunt folosite pentru a crea și edita elementele media individuale (text, grafică, sunet, video, animație) care alcătuiesc un produs multimedia.
 - **Authoring** applications conțin instrumente software pentru integrarea componentelor media și furnizarea unei interfețe cu utilizatorul.

Grafică

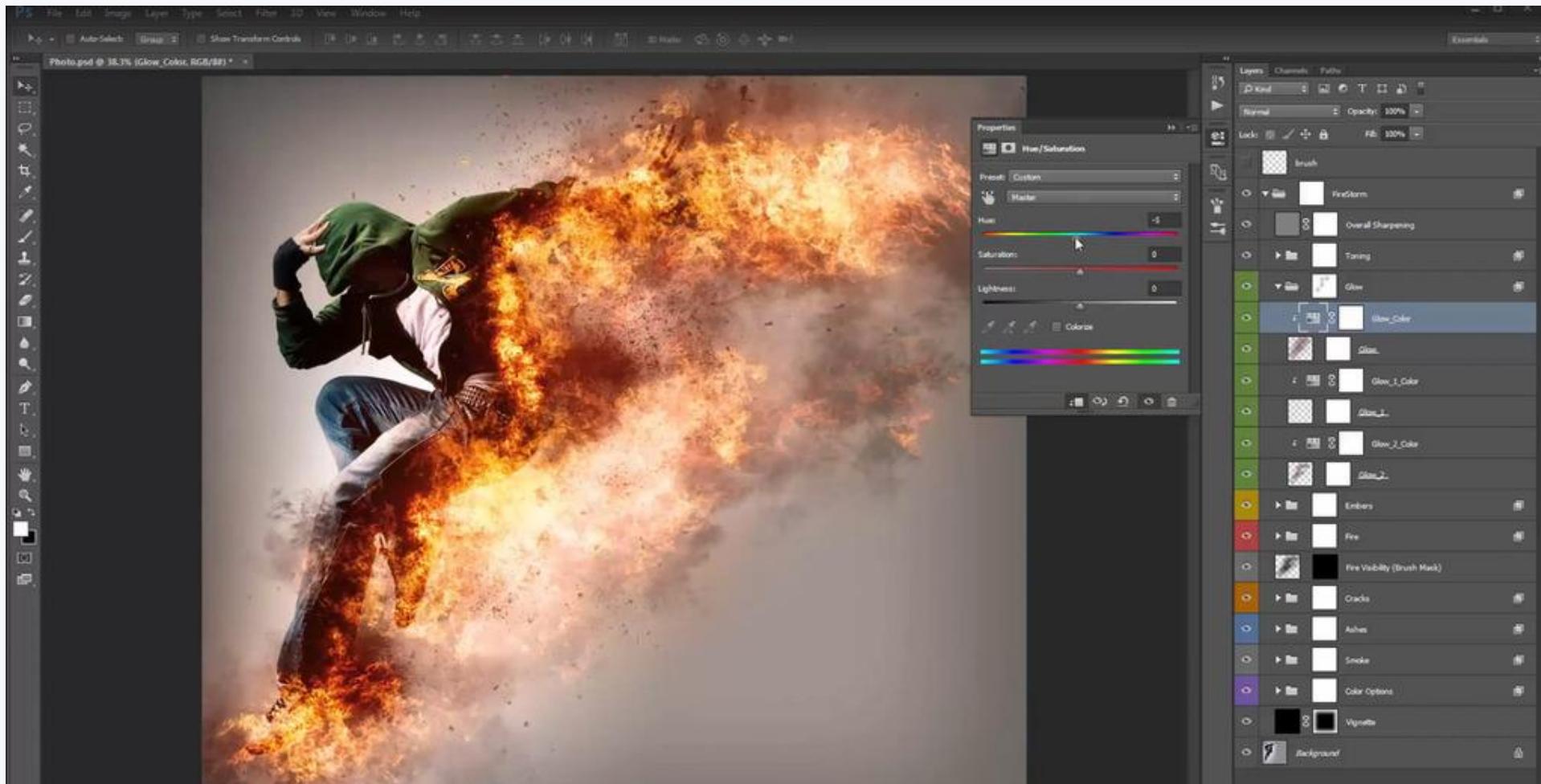
- Achiziția și prelucrarea de imagini 2D sau 3D.
- Categorii:
 - prelucrare imagini raster;
 - prelucrare imagini vectoriale;
 - 2D / 3D.

Aplicații prelucrare imagini raster

- conțin seturi de instrumente pentru a crea obiecte grafice, precum și instrumente de editare pentru fotografii digitale sau imagini scanate
- oferă o gamă largă de caracteristici, cum ar fi filtre (blur, emboss, pixelate), setări de reglare a imaginii (redimensionare, luminozitate, rotație) și efecte (umbră, gradient overlay).
- asigurarea unui control asupra elementelor individuale ale imaginii este posibilă folosind straturi (en: layers) și setări de mască. Instrumentele de tip text sunt utilizate pentru a genera text grafic cu modele, forme și efecte 3D.

Aplicații prelucrare imagini raster

- Example: Photoshop (Adobe), Gimp (open source)



Aplicații prelucrare imagini vectoriale

- conține un set distinct de instrumente pentru crearea de forme de bază, cum ar fi ovale, dreptunghiuri, curbe Bezier și poligoane generate utilizând formule matematice.
- formele pot fi grupate, umplute și redimensionate pentru a produce imagini complexe.
- astfel de aplicații pot fi folosite pentru a crea obiecte grafice care pot fi redimensionate cu ușurință pentru proiecte multimedia diverse.
- Example: Illustrator (Adobe), Draw (Corel), Inkscape (open source)

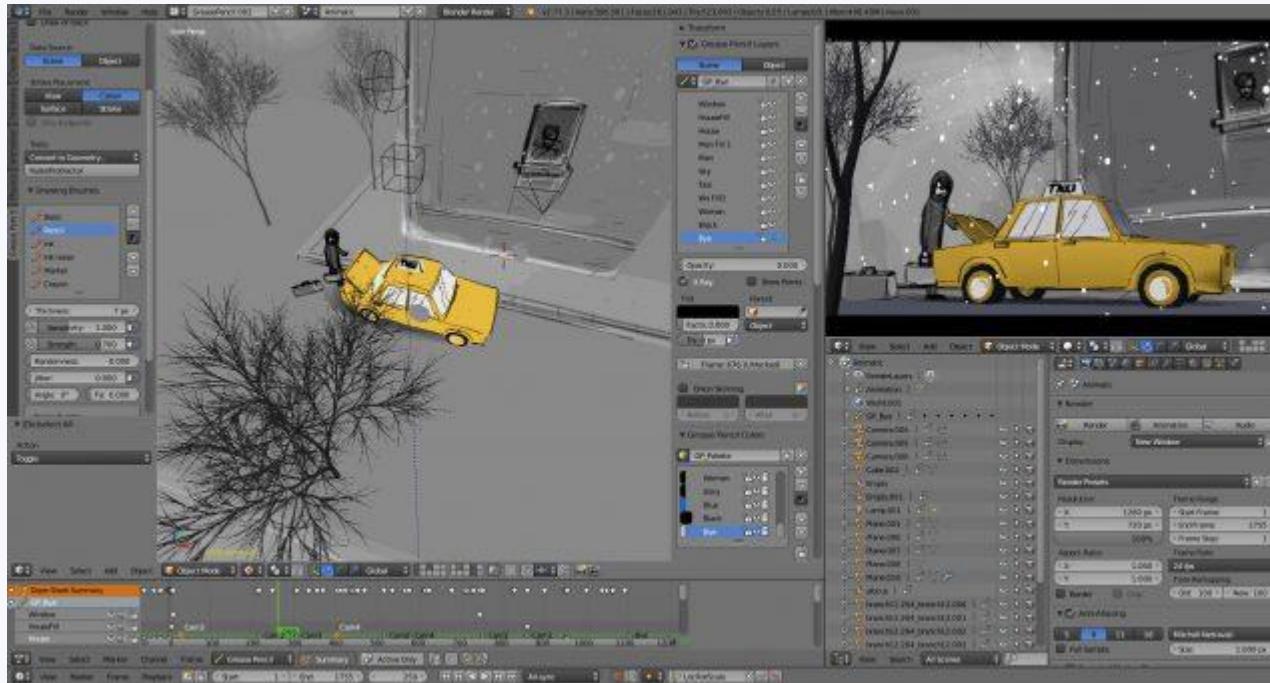
Aplicații prelucrare imagini vectoriale



Aplicații prelucrare imagini 3D

- folosit pentru a modela obiecte 3D, a defini suprafete, a compune scene și a exporta imaginea rezultată;
- În etapa de modelare (en: *modeling*), se creează forma obiectului; în etapa ulterioară, de definire a suprafetei (en: *surface definition*), se aplică culoarea și textura; în etapa de construire a scenei (en: *scene composition*), obiectele sunt aranjate în cadru, iluminate, se adaugă fundalul și efectele speciale. Ultima etapă, este reandarea (en: *rendering*). Randarea creează o imagine pornind de la o scenă 3D. Rendering is both processor intensive and time consuming because the software must calculate how the image should appear based on the object's position, surface materials, lighting, and specific render options.
- Example: Blender (open source)

Aplicații prelucrare imagini 3D



Sunet

- Există două tipuri majore de aplicații de sunet pentru dezvoltarea multimedia:
 - *sunet eșantionat* (en: *sampled*);
 - *sunet sintetizat* (en: *synthesized*).
- Example: Audition (Adobe)

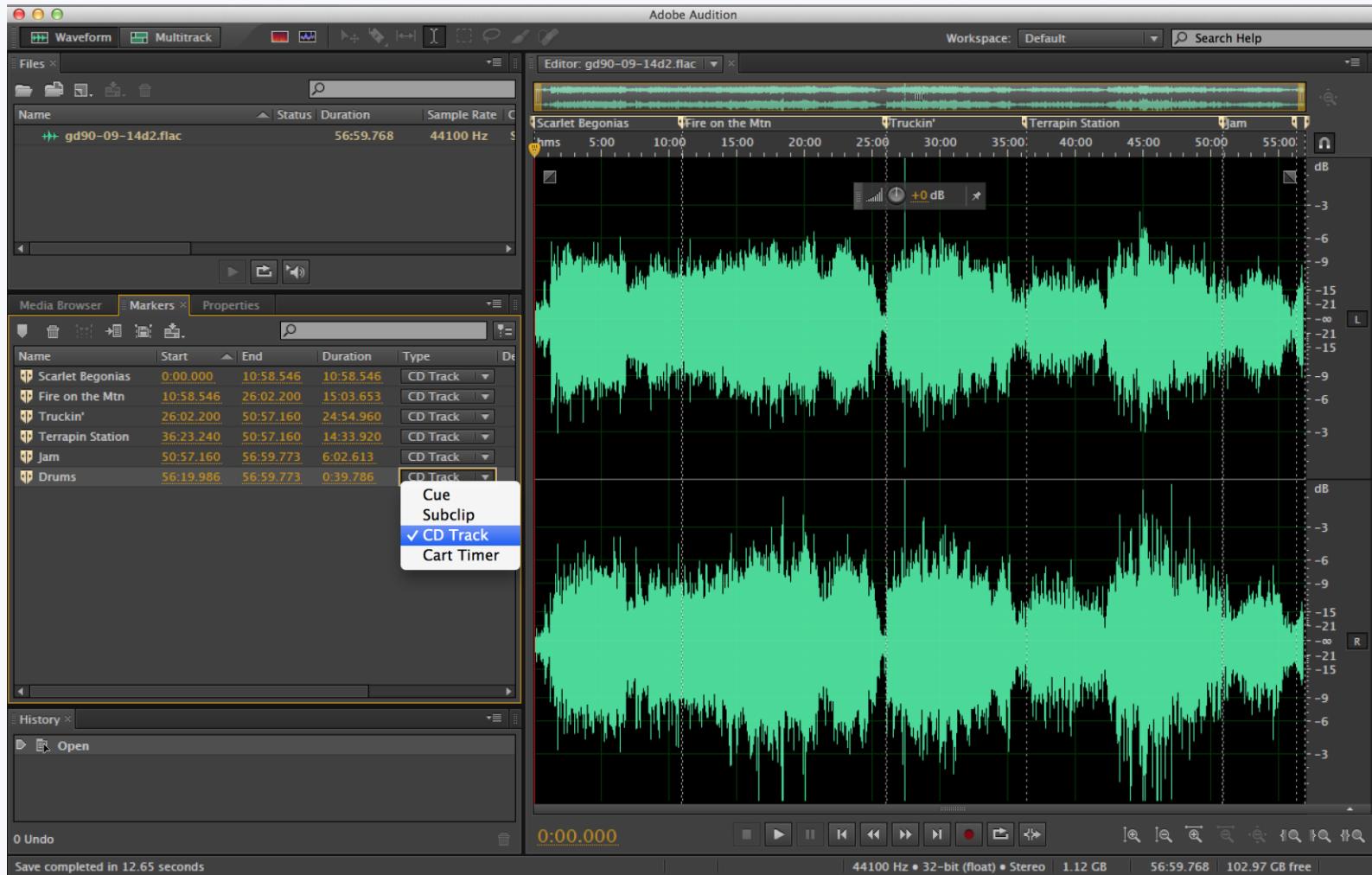
Sunet eşantionat

- sunetele eşantionate sunt reprezentări digitale ale surselor de sunet analogice captureate de la microfoane sau alte dispozitive.
- sunetele eşantionate pot fi editate într-o mare varietate de moduri, cum ar fi tăierea pentru a șterge porțiunile goale, îmbinarea pentru a combina segmente de sunet, setarea fade-in și fade-out, reglarea volumului și adăugarea de efecte speciale, cum ar fi ecouri sau inversări de sunet

Sunet sintetizat

- aplicațiile de sunet sintetizat folosesc comenzi digitale pentru a genera sunete. Aceste comenzi pot fi capturate de la un instrument MIDI, cum ar fi o tastatură electronică sau create cu un program specializat.

Software specializat Sunet



Video

- un mediu pentru a combina videoclipuri sursă, pentru a sincroniza clipurile cu o piesă sonoră, pentru a adăuga efecte speciale și pentru a salva rezultatul ca un videoclip digital.
- un proiect video începe prin încărcarea resurselor utilizate într-o fereastră de proiect. Resursele pot consta în imagini, animații, sunete, fișiere video.
- aplicațiile video oferă instrumente pentru a muta și insera clipuri pe o axă a timpului, pentru a tăia clipul și a defini tranzitii între piese. Aplicațiile de editare video definesc, de asemenea, dimensiunea redării și rata de afișare a cadrelor. Când proiectul video este finalizat, aplicația oferă setări pentru salvarea acestuia utilizând diferite scheme de compresie..

Software specializat Video

- Example: Premiere (Adobe)



Animație

- Folosit pentru a crea și edita secvențe animate. **Animația** este tehnica utilizării unei serii de imagini statice afișate rapid pentru a crea iluzia de mișcare.
- Fiecare cadru reprezintă o singură instanță a secvenței animate. Instrumentele tipice de animație controlează calea unui obiect, forma obiectului și modificările de culoare pe secvența de cadre.
- Obiectele sunt plasate pe o **axă a timpului** în care efectele pot fi aplicate pentru a se estompa, transforma, roti, răsturna sau schimba ritmul. Mai multe obiecte pot fi stratificate pentru a interacționa între ele pentru a crea animații mai complexe.

Software specializat

Animație

- Example: Director (Adobe), Flash (Adobe), Animate (Adobe)

Software pentru dezvoltare multimedia

- constă din programe special concepute pentru a facilita crearea de produse multimedia.
- acestea sunt utilizate pentru a combina elemente media, a sincroniza conținutul, a projecța interfața utilizatorului și a oferi interactivitate utilizatorului.

Software pentru dezvoltare multimedia

- Categorii în funcție de abordarea utilizată pentru a organiza elementele multimedia:
 - card-based;
 - timeline;
 - diagramă de flux.

Card-based

- elementele sunt aranjate la fel cum ar putea fi pe fișele de index sau pe paginile unei cărți.
- astfel de aplicații sunt ușor de utilizat, în special pentru prelegeri și tutoriale care nu necesită o sincronizare precisă a elementelor media individuale
- Example: **PowerPoint**, ToolBook

Timeline

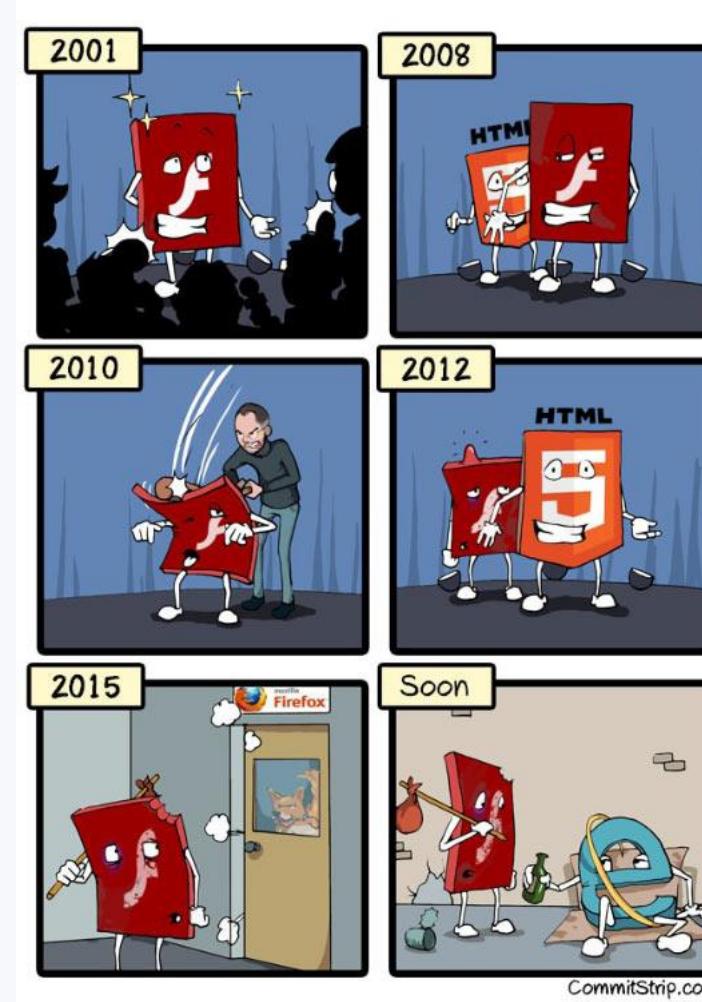
- Utilizarea unui *timeline* compus din cadre similar cu un film;
- astfel de aplicații oferă controlul precis necesar pentru animații avansate;
- Example: Director (Adobe), Flash (Adobe), Animate(Adobe).

Software pentru dezvoltare multimedia

Adobe Flash



Evoluția Adobe Flash



Diagramă de flux

- utilizează diagrame de flux pentru a dezvolta rapid o gamă largă de produse multimedia, inclusiv tutoriale avansate, demonstrații de produse și simulări. Pictogramele pot reprezenta atât conținut (imagini, text, animații, video), cât și o gamă largă de interacțiuni (redare, oprire, accesare, calculare etc.).
- example: Authorware

Multimedia în context WEB

Multimedia în context WEB

- HTML5 oferă un suport mult îmbunătățit pentru multimedia web, prin includerea elementelor precum audio, video, canvas
- elemente multimedia suportate:
 - text
 - imagini
 - animație
 - grafică raster - elementul <canvas>
 - grafică vectorială – elemental <svg>
 - 3D graphics - WebGL
 - audio – elementul <audio>
 - video – elementul <video>

HTML5 Games



http://www.cuttherope.net/basic_standalone/game.html

HTML5 Videos



Limbajul JavaScript

Caracteristici

- Dinamic
- Bazat pe obiecte

Utilizare

- Manipulare noduri DOM
- Evenimente
- Procesare
- Comunicare la distanță

Modalități de includere în HTML

Prin intermediul tag-ului <script>:

1. Fișier extern

```
<script type="text/javascript" src="lib/test.js"></script>
```

2. În cadrul paginii

```
<script>
// cod JavaScript ....
let test = 10;
</script>
```

Tipuri de date

Tipuri de date de bază

- Number: -3.14, 6, 2
- Boolean: *true, false*
- String: "test"
- *null, undefined*
- Object: { nume: "Ana", varsta:7 }

Further reading: https://www.w3schools.com/js/js_datatypes.asp

Tipuri de date

Obiecte speciale

- Function: *function f() {...}*
- Array: [1, 2, "trei"]

Variabile și expresii

Declarare

- Prin atribuire valoare (nerecomandat): `a = 8; a = false;`
- Folosind `var`, `let` sau `const`:

```
var a = 3; var b;
let b = "test"; let b;
const b = "test"
```

Exemplu:

```
const c = 7; // c = 3; //error
const v = [1, 2, 3]; v[0] = 5; //works fine
```

Variabile și expresii

Scope:

- Global sau local
- Function vs block based

Evaluare expresii

- operatori similari cu C# (în plus `==`, `!=`)

Exemple:

```
a += 7; a++;  
a < b && b > c; a > 10;  
a = "Ana" + "-" + "Maria";  
"10" == 10; "10" === 10;
```

Vectori – Obiectul *Array*

Initializare: `var v = [];` sau `var v = [1, "Ion"]`

Accesare elemente: `var i = v[0]; v[1] = 23;`

Dimensiune: `v.length` (*read / write*)

Metode:

- `push(valoare)` – adăugare la sfărșit
- `pop()` – extrage ultimul element
- `indexOf(valoare)` – întoarce poziția elementului (sau -1)
- `sort()` – sortează vectorul
- `slice(index_start, index_sfârșit)` – extrage un sub-vector

Functii

Declarare:

function suma(a, b) { return a + b; }

var suma = function(a, b) { return a + b; }

Parametri:

Transmisi prin valoare

Accesibili prin *arguments*

Apel - nume_functie / expresie(parametri):

var rezultat = suma(7, 3);

var test = function(val) { return val + 1; }(5);

Obiecte

Obiect = colecție de proprietăți (perechi *nume* = *valoare*)

Declarare obiecte

Literali: var ob = { nume: "Ana", varsta:7 }

new: let ob = new Object();

Accesare proprietăți

Citire: var v = ob.varsta; sau var v = ob["varsta"]; for (v in ob) {...}

Modificare / adăugare: ob.varsta = 8; ob["varsta"] = 8; ob.clasa = 2;

Obiecte

Metode

adăugare:

```
ob.test = function() { console.log("test"); }
```

this:

```
ob.afisare = function() { console.log("Nume: " + this.nume); }
```

Apel:

```
ob.afisare();
```

Constructori și moștenire

Functiile JavaScript pot fi utilizate pentru construirea de obiecte

Exemplu: *function Persoana(ume) { this.ume = nume; this.varsta = 1; }*

Apel: *let ob2 = new Persoana("Maria");*

prototype

Proprietate a functiei constructor

Definește proprietățile disponibile în toate obiectele (instanțele create)

Exemplu: *Persoana.prototype.afisare = function() { console.log("Nume: " + this.ume); };*

Folosit și pentru implementarea moștenirii:

```
function Student() { this.facultate = "CSIE"; }
```

```
Student.prototype = new Persoana("-");
```

```
var s = new Student(); s.afisare();
```

DOM - Structura

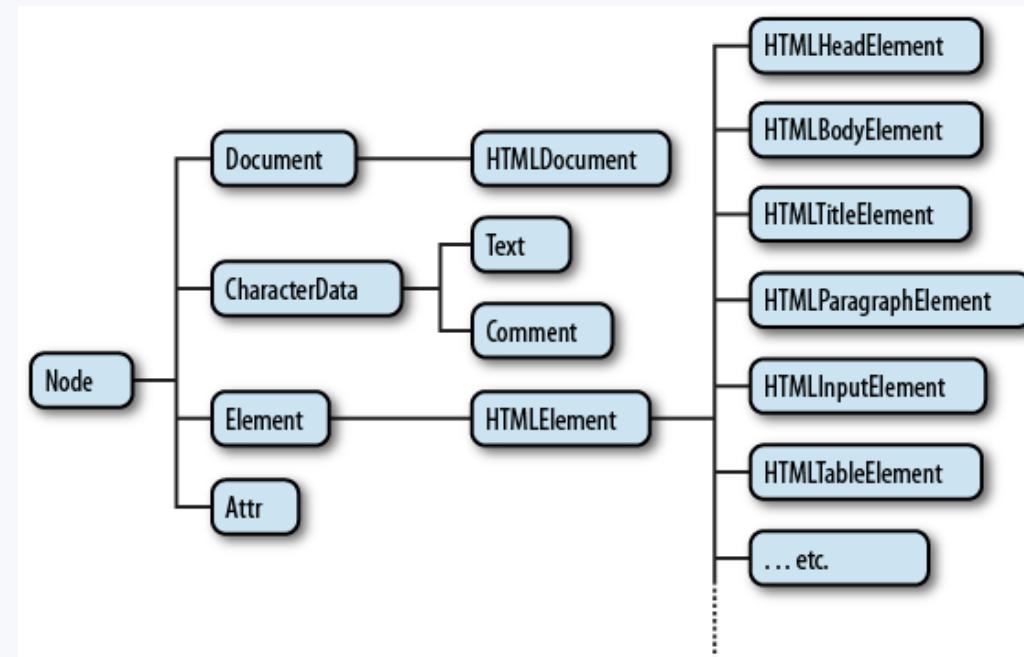
Arbore construit din obiecte JavaScript

Fiecare nod:

- Este un obiect derivat din *Node*
- Conține o colecție de referințe către nodurile copil: *childNodes / children*
- Conține referințe către nodul părinte (*parentNode*) și nodul următor (*nextSibling*)
- Conține metode pentru manipularea nodurilor copil: *appendChild(nod)*, *removeChild(nod)*, ...

Rădăcina: *document.documentElement*

Nodurile au metode și proprietăți specifice în funcție de tag-ul HTML utilizat pentru construirea nodului



Sursa: <http://web.stanford.edu/class/cs98si/slides/the-document-object-model.html>

Regăsire noduri

Pe bază de identificator

```
elem = document.getElementById(identificator)
```

Pe bază de selector CSS

```
elem = element.querySelector("selector CSS");
```

```
lista = element.querySelectorAll("selector CSS");
```

Alte metode

```
lista = element.getElementsByClassName("clasa CSS");
```

```
lista = element.getElementsByTagName("tag HTML");
```

Manipulare noduri

Construire nod:

```
elem = document.createElement("tag HTML");
```

Accesare atribut:

elem.numeAtribut – accesare valoare atribut individual

elem.attributes – colecția de atribut

Accesare atribut CSS:

elem.style.numeAtributCSS

Exemplu:

```
var p = document.createElement("p");
p.innerText = "test";
document.querySelector("body").appendChild(p);
p.style.color = "red";
```

Tratare evenimente

Adăugare event handler:

- Prin atribut HTML:

```
<element atributEveniment="nume funcție">...</element>
```

Exemplu: `test`

- Prin proprietăți nod:

```
element.proprietateEveniment = funcție;
```

Exemplu:

```
let elem = document.getElementById("test");
elem.onclick = function() {console.log("un mesaj");}
```

Tratare evenimente

- Prin metoda addEventListener:

element.addEventListener(tipEveniment, funcție);

Exemplu:

```
document.getElementById("test")
    .addEventListener(
        "click",
        function() {console.log("un mesaj");}
    );
```

Eliminare event handler:

```
element.removeEventListener(tipEveniment, funcție);
```

Tratare evenimente

Accesare element sursă – *this*

Parametri eveniment – obiect *event*

- General: *target*, *type*, *preventDefault()*
- Tastatură: *key*, *keyCode*, *altKey*, *ctrlKey*, *shiftKey*
- Mouse: *pageX*, *pageY*, *button*, *altKey*, *ctrlKey*, *shiftKey*

Evenimente

- General: *load*
- Tastatură: *keydown*, *keypress*, *keyup*
- Mouse: *mouseenter*, *mouseleave*, *mousemove*, *mouseup*, *mousedown*, *click*, *dblclick*

Programare execuție funcție

A) Execuție o singură dată după un interval de timp specificat

Programare pornire:

var id = setTimeout(funcție, durata[, param1, param2, ...]);

Oprire:

clearTimeout(id);

B) Execuție repetată la un interval de timp fix

Programare pornire:

var id = setInterval(funcție, durata[, param1, param2, ...]);

Oprire:

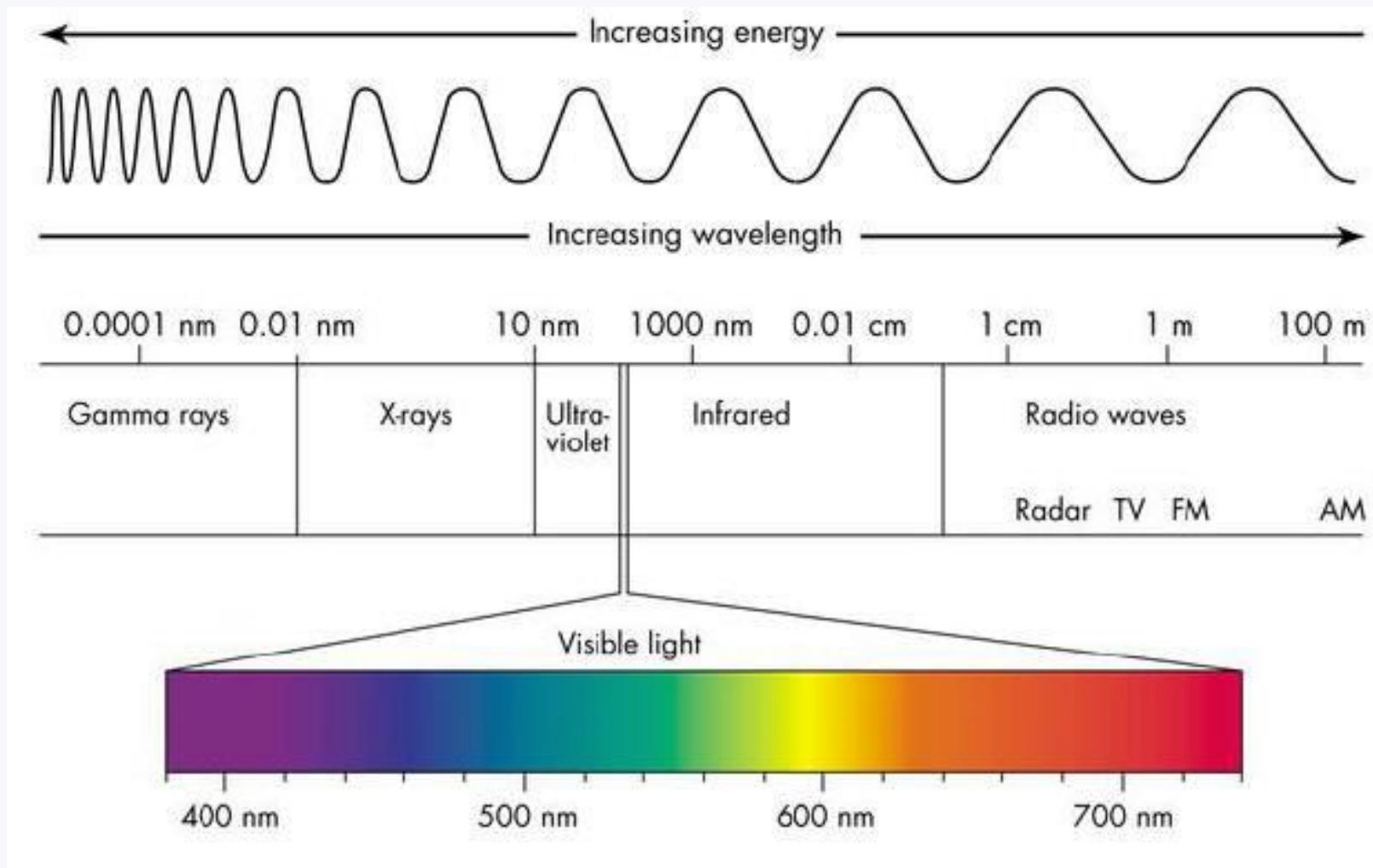
clearInterval(id);

Observație: duratele sunt exprimate în milisecunde

JavaScript

- <http://jstherightway.org/>

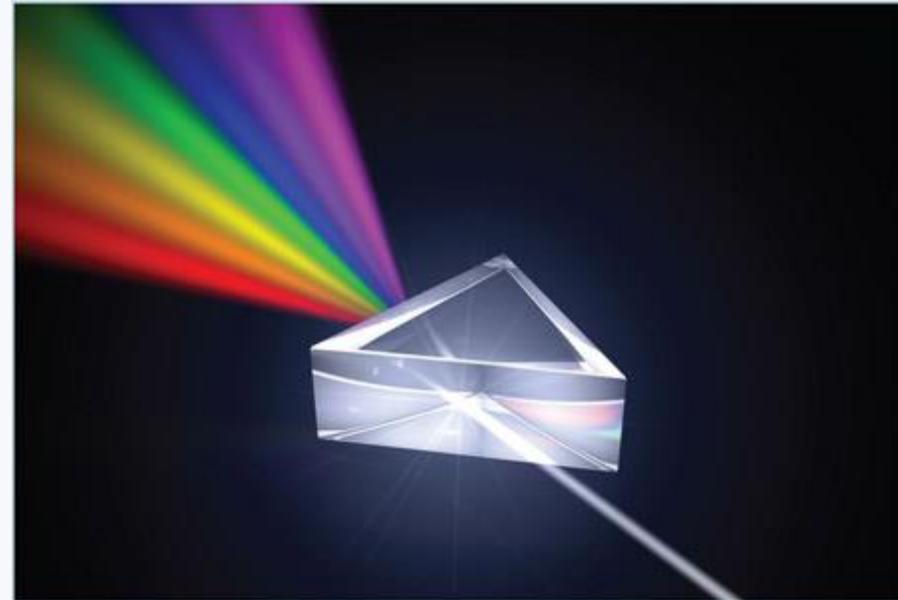
Imaginea



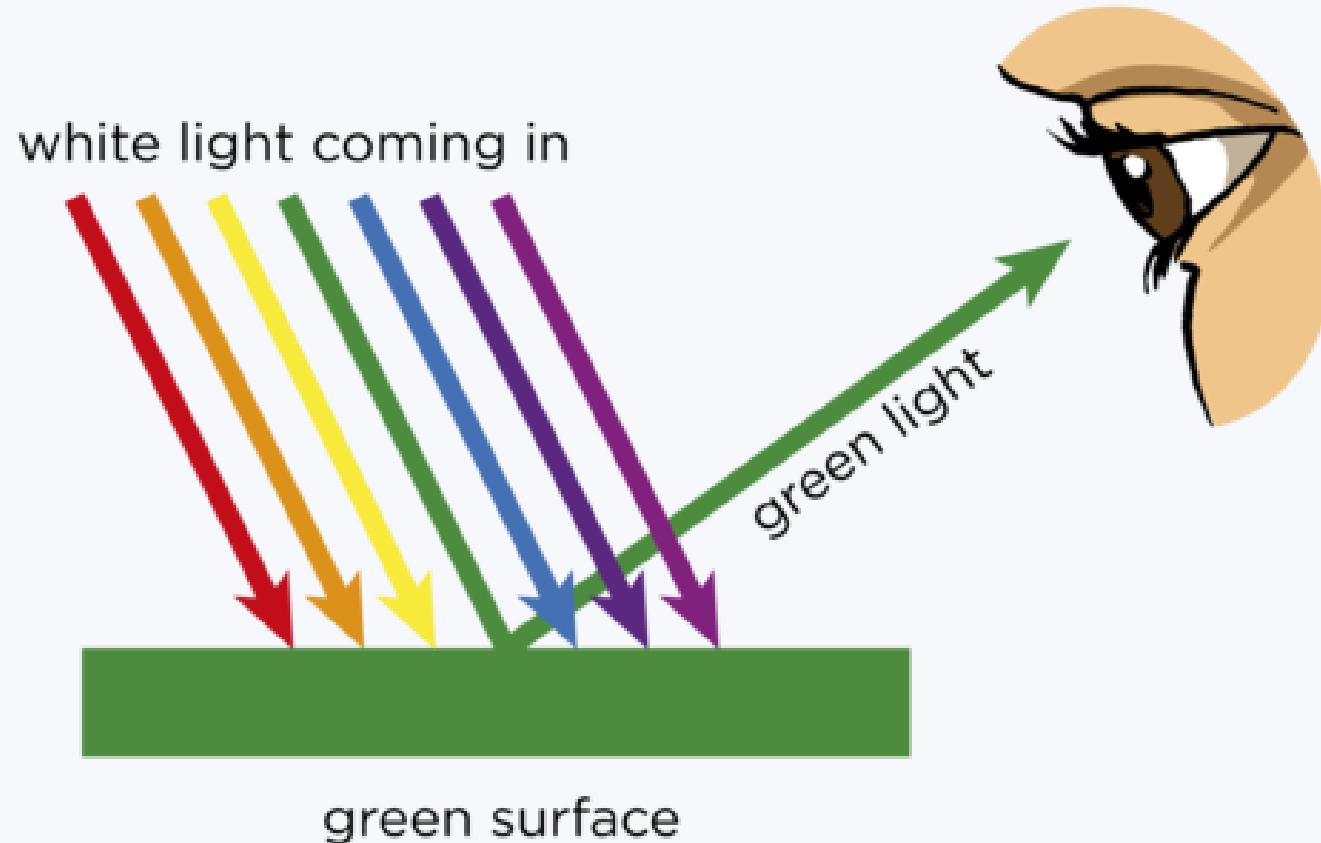
Lumina

- ne referim la lumina soarelui naturală ca lumină albă, deoarece ochiul uman o percepă ca fiind incoloră.
- este posibilă separarea luminii albe cu ajutorul unei prisme. La trecerea luminii printr-o prismă, aceasta este descompusă în lungimile de undă ale culorii sale componente.

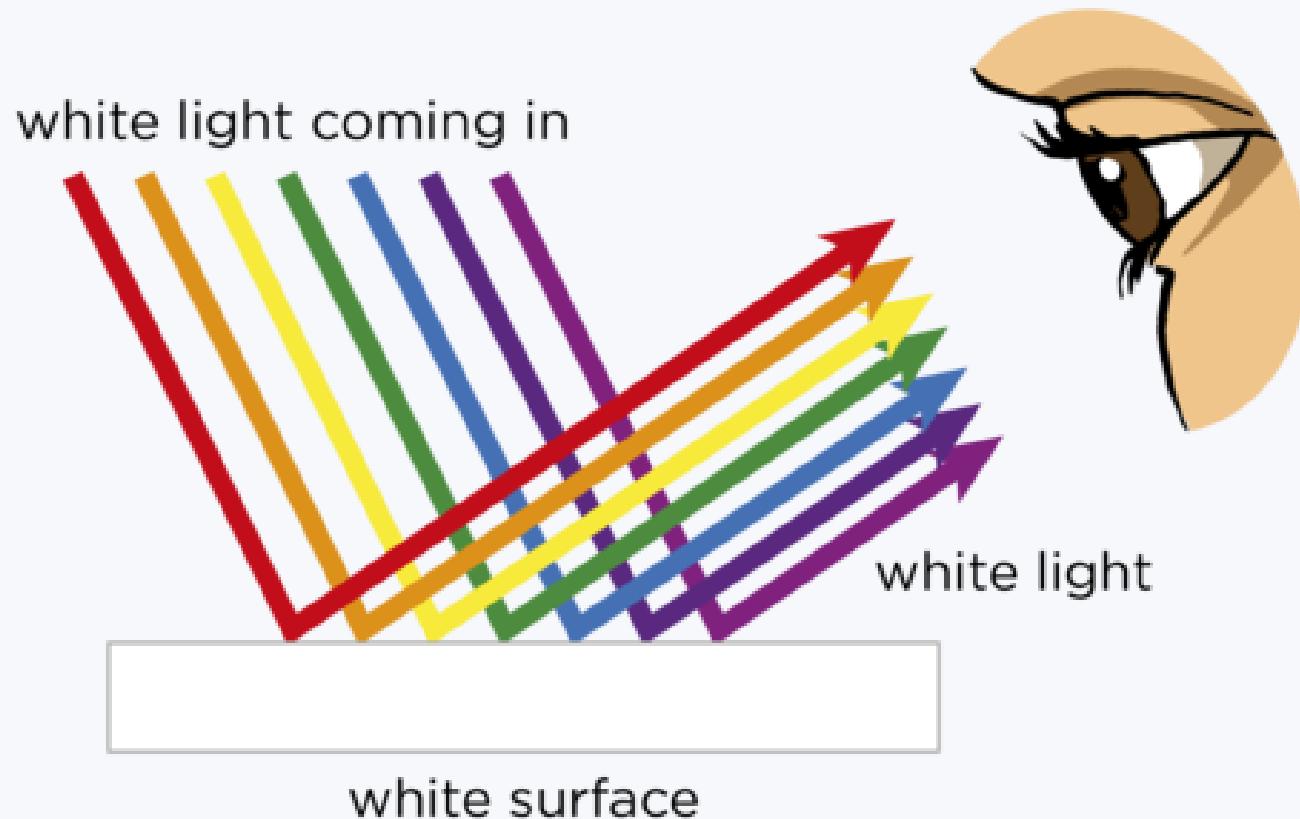
Imaginea Lumina



Culorile primare și secundare ale luminii albe devin vizibile atunci când sunt refractate de o prismă de sticlă.



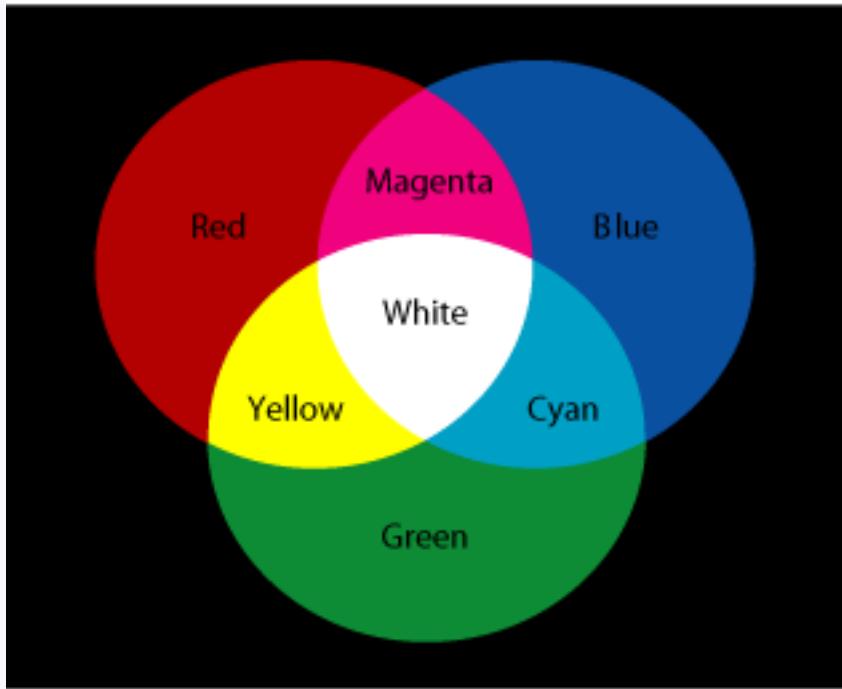
O suprafață verde absoarbe toate culorile, cu excepția verdelui, pe care îl reflectă.



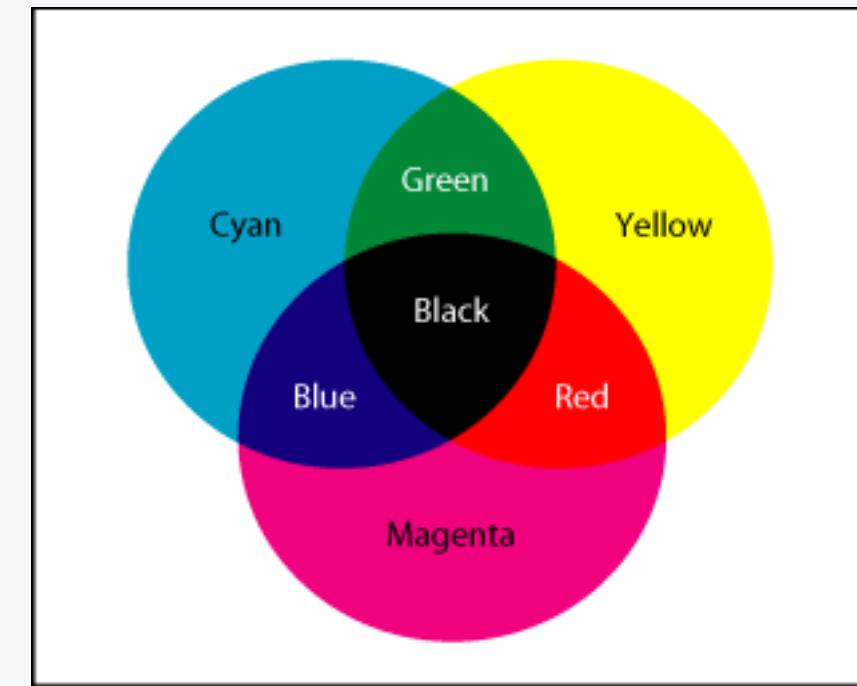
Lumina

- Deși putem obține vopsea neagră ca pigment, negrul nu este o culoare a luminii. Negrul este rezultatul absorbției complete a luminii.

Modele de culoare



Model aditiv(RGB)



Model subtractiv (CMYK)

Model aditiv

- În **modelele aditive** culorile sunt create prin amestecarea mai multor culori primare.
- **Roșu, verde și albastru** sunt culorile primare cel mai des utilizate în modelul aditiv.
- Modelul aditiv pornește de la **negru** și ajunge la **alb**; pe măsură ce se adaugă mai multă culoare, rezultatul este mai deschis și tinde spre **alb**.
- Combinarea a **două** dintre cele **trei** culori primare standard utilizate în modelul aditiv în proporții egale produce una dintre culorile secundare din modelul aditiv - **cian, magenta sau galben**

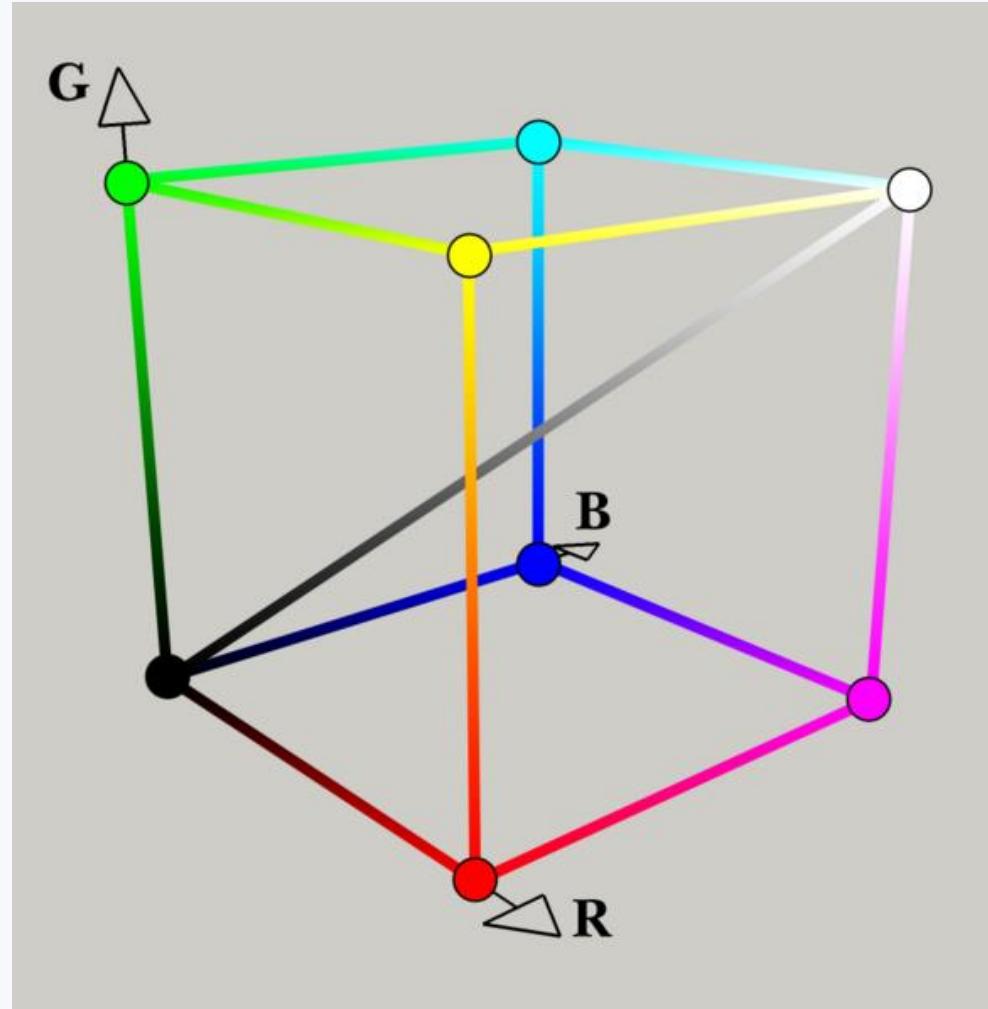
Modelul RGB

- utilizează culorile de bază **roșu, verde și albastru (RGB)**.
- reglând intensitatea fiecărei culori se pot obține toate culorile din spectrul luminii vizibile. Se va obține culoarea albă prin combinarea culorilor de bază în mod egal.
- Dacă ar fi să ne uităm la un monitor LCD (ecran cu cristale lichide) la microscop, am vedea că fiecare pixel afișează în realitate doar cele trei culori primare ale luminii.

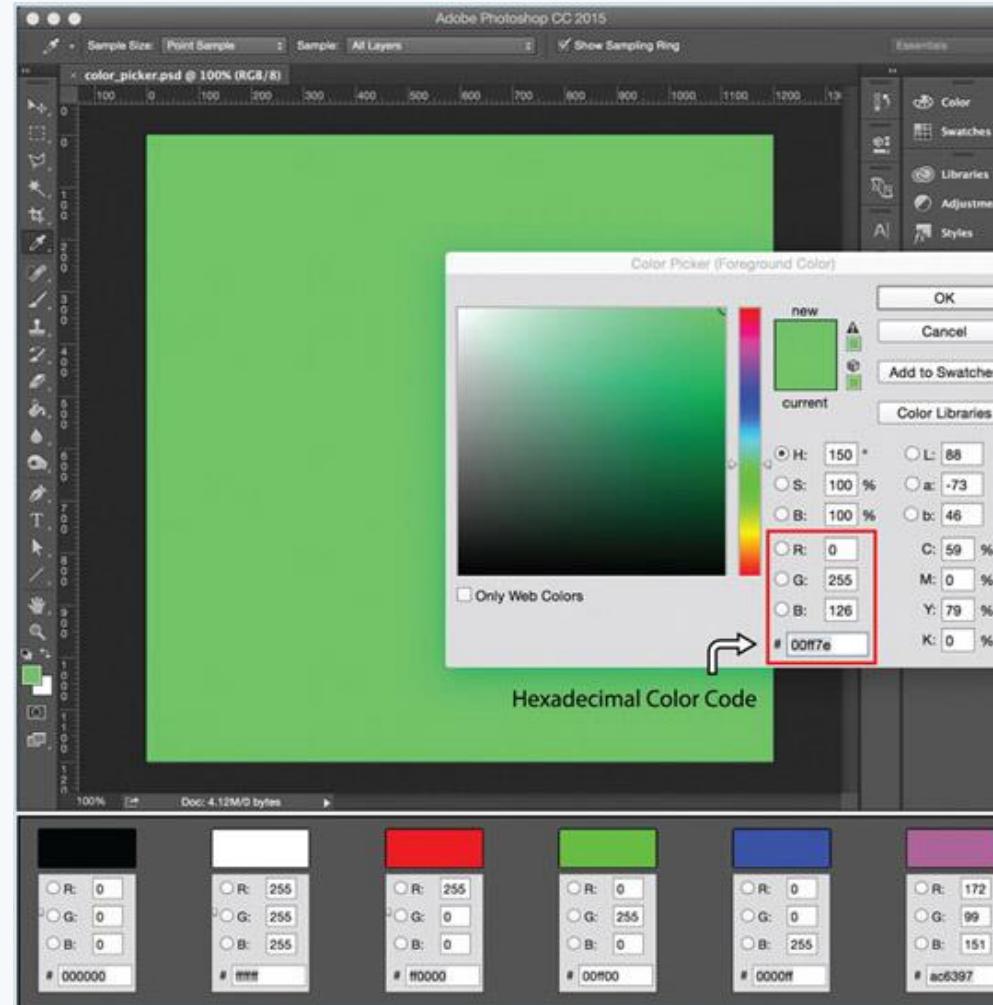
Modelul RGB

- Prin combinarea culorilor roșu și verde se obține culoarea galben. Dacă umplem o formă geometrică cu galben în Photoshop, pixelii aferenți zonei respective vor avea active doar componente de roșu și verde (componenta de albastru nu va fi activă).
- Punctele individuale de culoare sunt în cazul ecranelor LCD suficient de mici, încât creierul uman combină culorile și le percep ca galben.

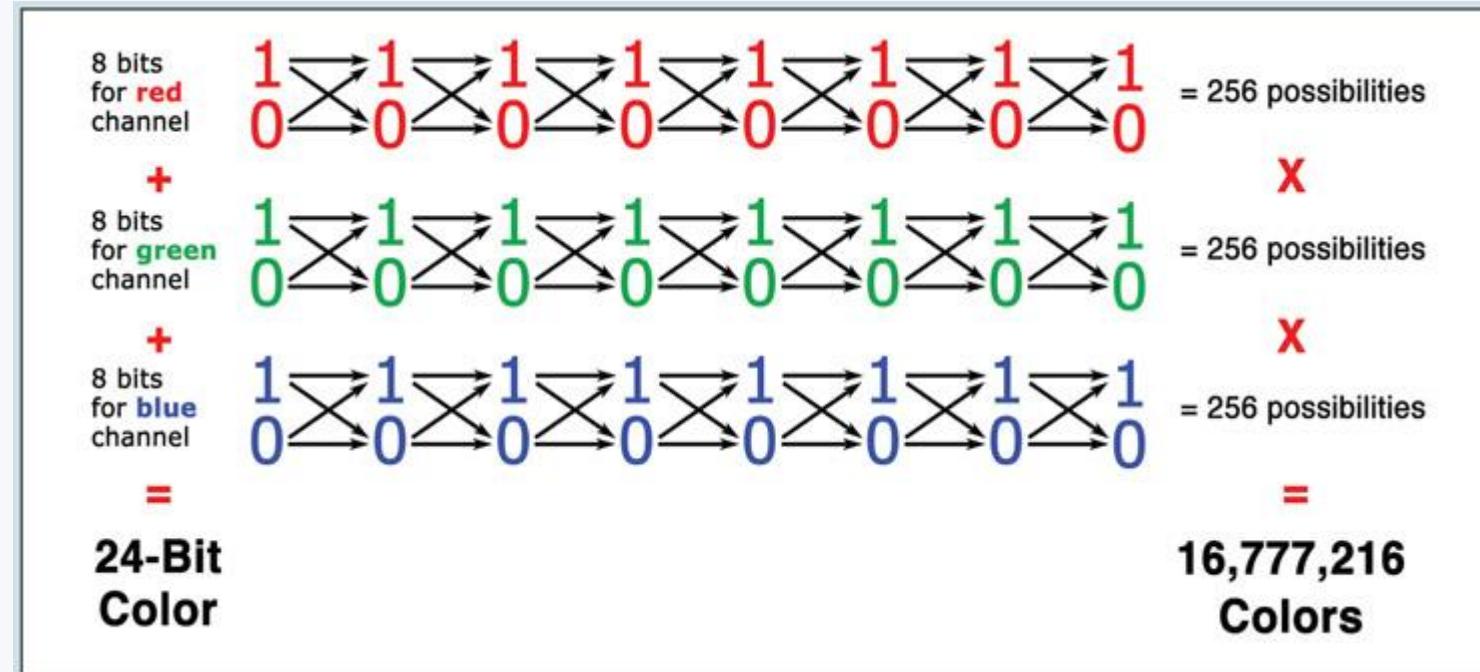
Modelul RGB



Adobe Photoshop – Modelul RGB



Modelul RGB



Combinăriile de culori posibile pentru un pixel, utilizând 8 biți pentru fiecare culoare (afișaj pe 24 de biți).

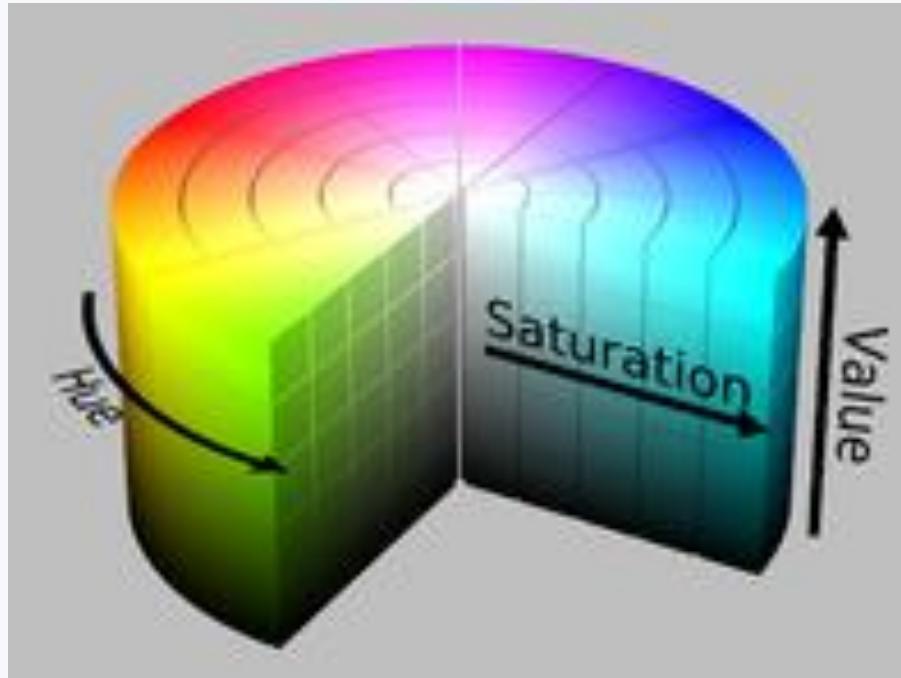
Modelul RGB

- 256 de valori posibile pentru fiecare canal de culoare.
- 256 de valori posibile pentru roșu × 256 pentru verde × 256 pentru albastru - **16.8 milioane** de combinații posibile / culori.

HSL - Hue, Saturation and Lightness Color Model

- Reprezentare sub formă de coordonate cilindrice
- Bazat pe aceleași culori de bază:
 - Roșu - 0^0
 - Verde - 120^0
 - Albastru - 240^0
- Cele două reprezentări rearanjează geometria RGB într-o încercare de a o face mai intuitivă și mai relevantă perceptual decât reprezentarea carteziană (cub)

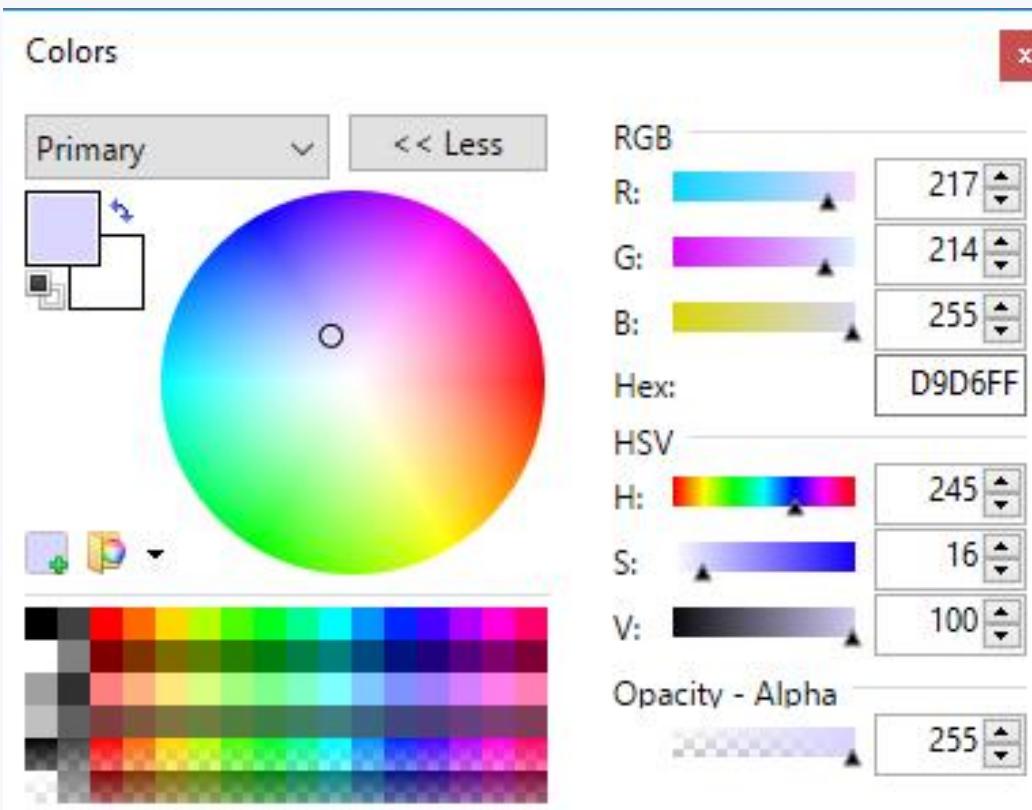
HSL - Hue, Saturation and Lightness Color Model



- Roșu - 0^0
- Verde - 120^0
- Albastru - 240^0

HSL - Hue, Saturation and Lightness Color Model

- HSL și HSV sunt utilizate în instrumentele de selectare a culorii, în analiza imaginilor și computer vision.



Instrument de selecție a culorii în Paint.NET:
<http://www.getpaint.net/index.html>

Modele subtractive

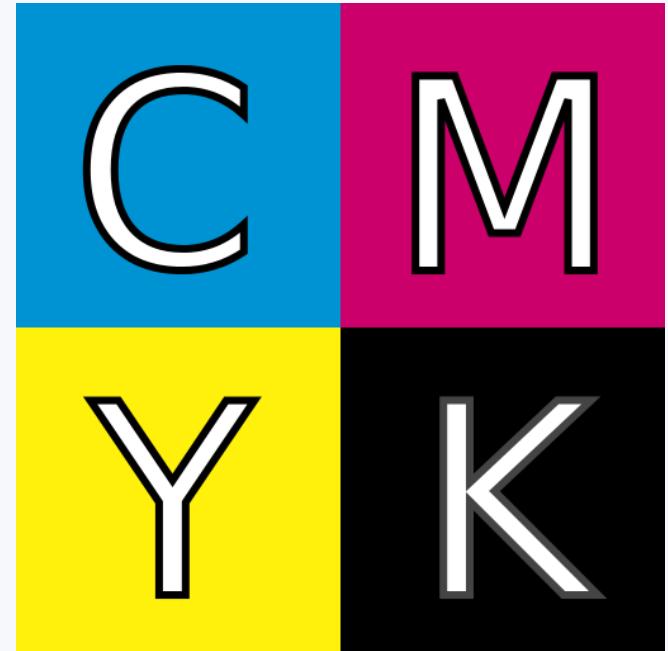
- Combinarea culorilor în procesul de imprimare utilizează metoda **subtractivă** [1].
- Modelul **subtractiv** explică cum este posibil ca prin combinarea unui număr limitat de pigmenți sau coloranți naturali să se obțină o paletă largă de culori. Adăugarea fiecărei culori determină reducerea parțială sau completă (absorbția) unor lungimi de undă a luminii (și reflectarea celorlalte)
- În acest model se pornește de la culoarea albă și se ajunge la negru; pe măsură ce se adaugă culoare, rezultatul devine mai întunecat și tinde spre negru [2].

[1] https://en.wikipedia.org/wiki/Subtractive_color

[2] http://www.worqx.com/color/color_systems.htm

Modelul CMYK

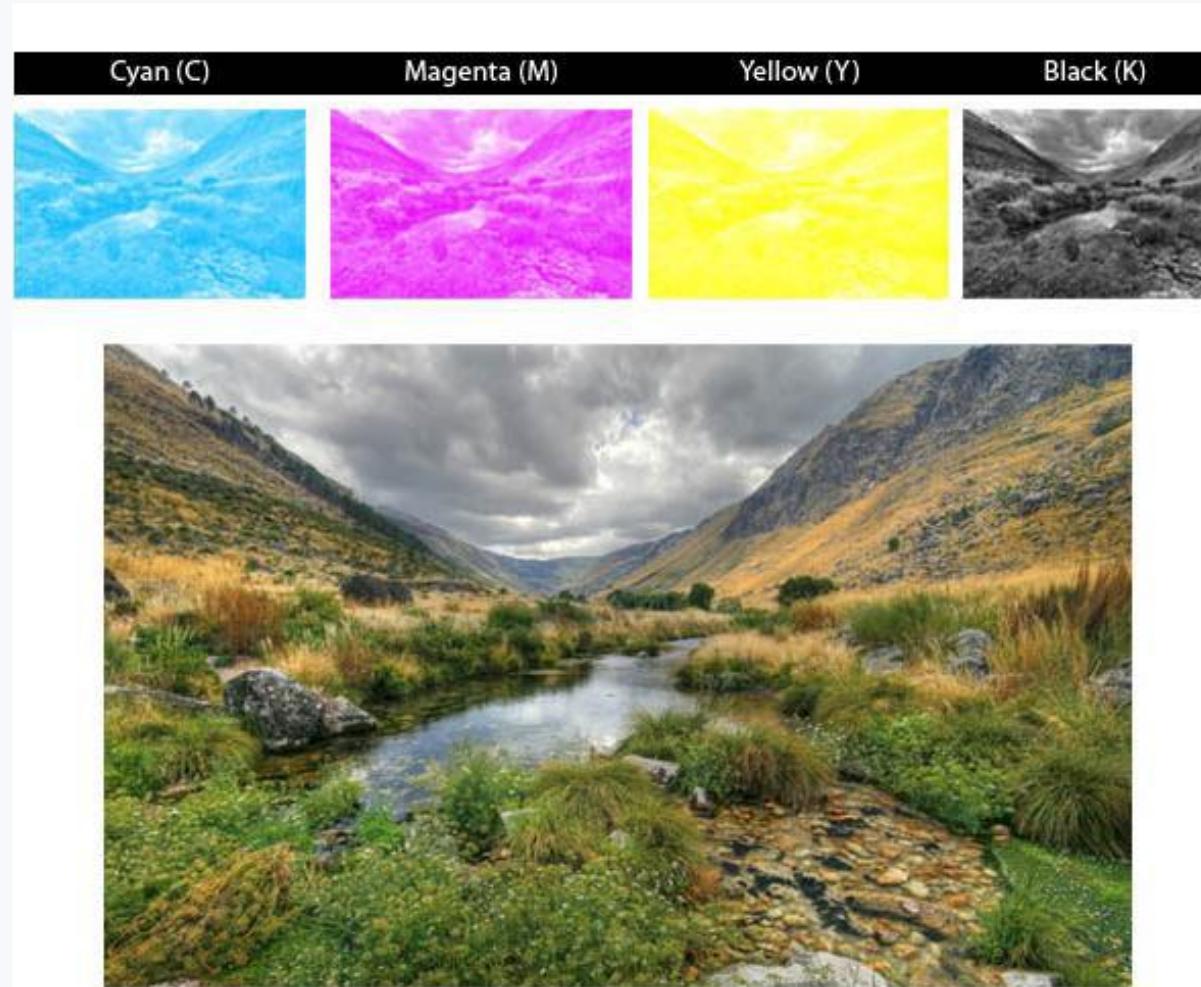
- Modelul de culoare CMYK este utilizat în procesul de imprimare[1].
- CMYK se referă la cele patru cerneluri utilizate în majoritatea imprimărilor color: cyan, magenta, yellow și key (black) [1].
- Deși variază în funcție de tipografie, producător, imprimantă, cerneala este de obicei aplicată în ordinea abrevierii[1].



Modelul CMYK

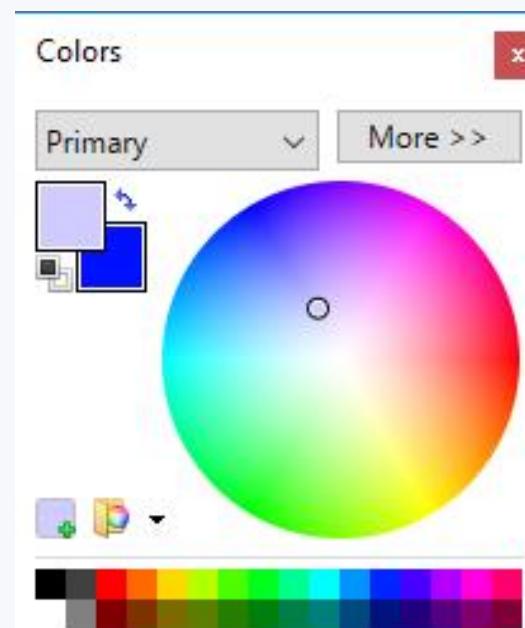


Modelul CMYK



Cercul culorilor (en: Color Wheel)

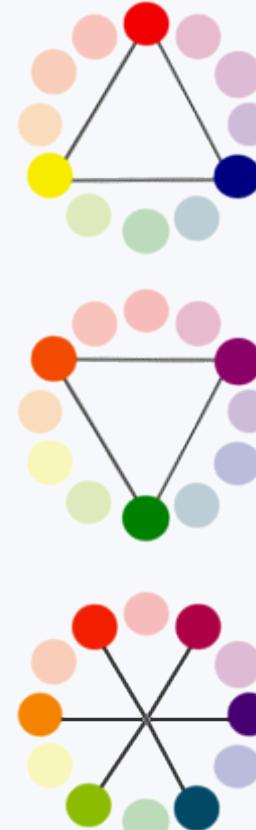
- Cercul culorilor este o reprezentare vizuală a culorilor dispuse în funcție de **relațiile cromatice** dintre ele. Cercul culorilor este format prin poziționarea nuanțelor primare echidistant una față de cealaltă. Între culorile primare sunt apoi plasate culorile secundare și terțiare [1].



Color Picker in Paint.NET:
<http://www.getpaint.net/index.html>

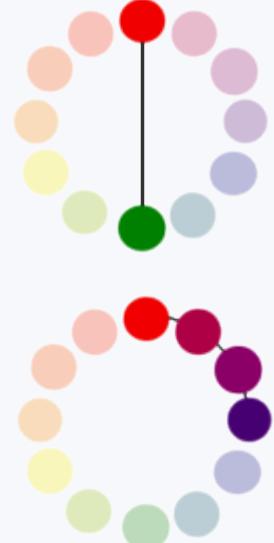
Cercul culorilor (en: Color Wheel)

- **Culori primare:** culorile care nu pot fi create prin amestecarea altora[1].
- **Culori secundare:** acele culori realizate printr-un amestec de două culori primare [1].
- **Culori terțiare:** acele culori realizate printr-un amestec de nuanțe primare și secundare[1].



Cercul culorilor (en: Color Wheel)

- **Culori complementare:** acele culori situate diametral opus pe cercul culorilor [1].
- **Culori apropiate:** acele culori situate aproape una de cealaltă pe o cercul culorilor[1].



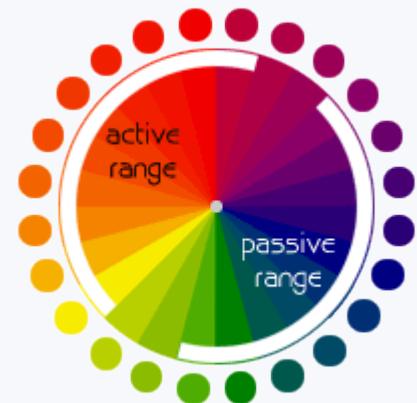
Active & Passive Colors

- The color wheel can be divided into ranges that are visually:
 - Active colors - will appear to advance when placed against passive hues.
 - Passive colors - appear to recede when positioned against active hues.
- Advancing hues are most often thought to have less visual weight than the receding hues.
- Most often warm, saturated, light value hues are "active" and visually advance.



Active & Passive Colors

- Cool, low saturated, dark value hues are "passive" and visually recede.
- Tints or hues with a low saturation appear lighter than shades or highly saturated colors.
- Some colors remain visually neutral or indifferent.



Complementary Colors

- When fully saturated complements are brought together, interesting effects are noticeable. This may be a desirable illusion, or a problem if creating visuals that are to be read.



Does this text have
highlighted edges?

- Notice the illusion of highlighted edges and raised text. This may occur when opposing colors are brought together.

CSS – Specificarea colorilor

Format hexazecimal:

- $\#rrggbb$
- rr, gg, bb sunt numere în baza 16

```
<h1 style="background-color:#ff6347;">...</h1>
```

Format RGB:

- $rgb(red, green, blue)$ sau $rgba(red, green, blue, alpha)$
- $red, green, blue$ – numere de la 0 la 255 sau procente
- $alpha$ – număr între 0 – transparent și 1 – opac sau procent

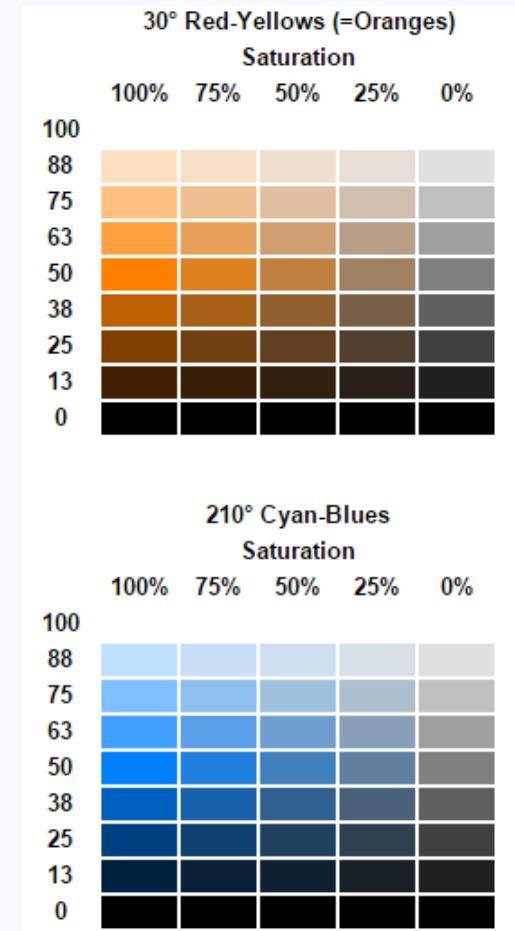
```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
```

CSS – Specificarea colorilor

Format HSL:

- *hsl(hue, saturation, lightness)* sau *hsla(hue, saturation, lightness, alpha)*
- *hue* – număr de la 0 la 360
- *saturation, lightness* – procente
- *alpha* – număr între 0 – transparent și 1 – opac sau procent

```
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>
```

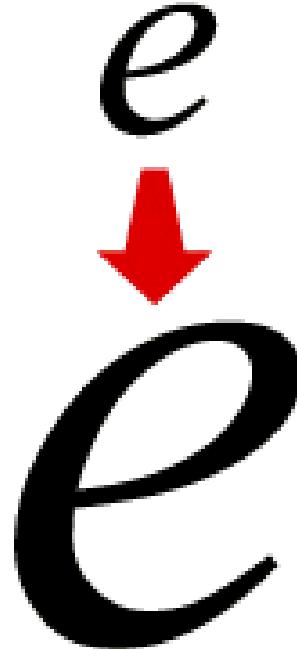
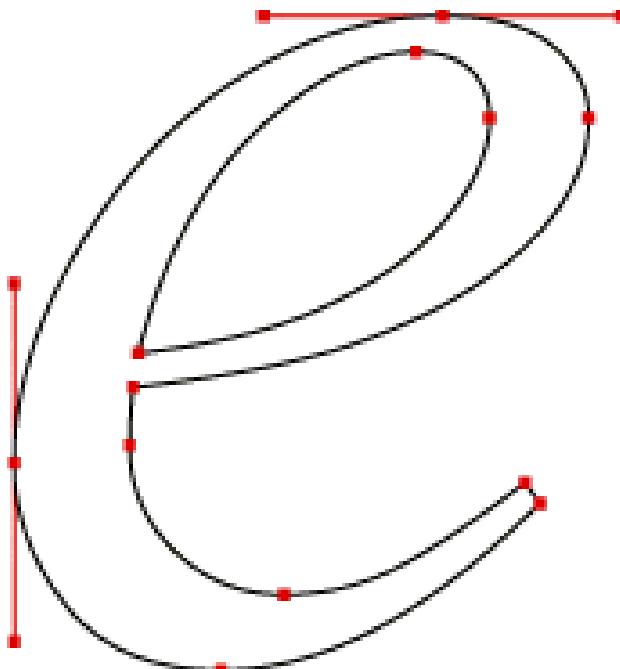


Grafică pe computer 2-D

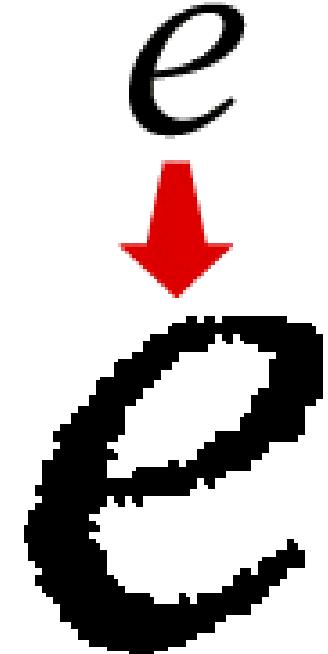
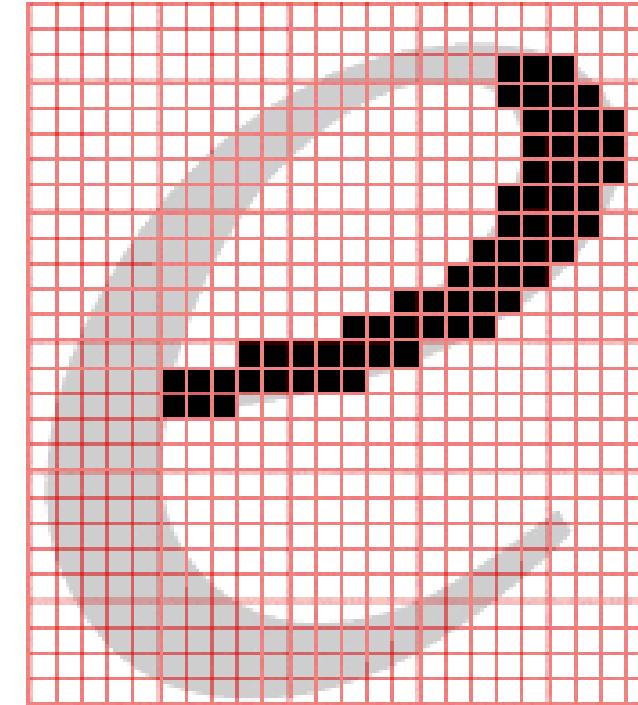
- Calculatoarele pot fi utilizate pentru a crea imagini 2D (lățime și înăltime) sau 3D (lățime, înăltime și adâncime).
- Există două tipuri principale de grafică computerizată 2D:
 - **bitmap** - potrivită pentru imagini cu detalii fine, cum ar fi picturi sau fotografii. Numită și **grafică raster**. Stochează informația despre culoarea fiecărui pixel.
 - **vectorială** - utilizat pentru desene grafice, de la desene și logo-uri simple la creații artistice sofisticate. Utilizează descrieri matematice.

Grafică pe computer 2-D

VECTOR GRAPHICS



BITMAPPED (RASTER) GRAPHICS



Grafică raster

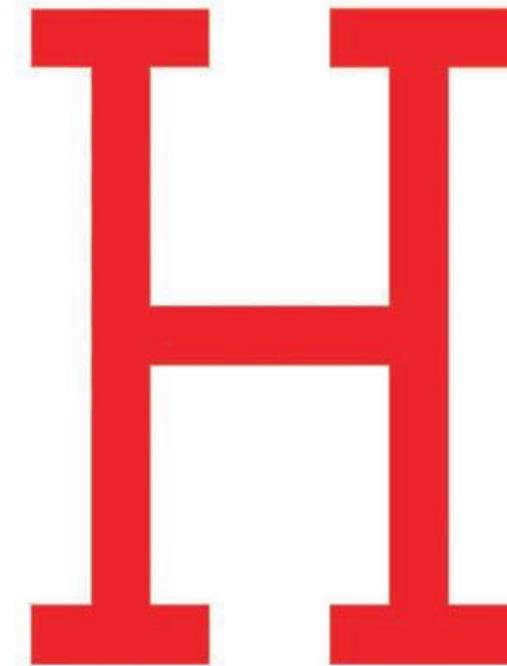
Imagini digitale

- O **imagine** este o reprezentare bidimensională sau tridimensională a unei persoane, obiect sau scenă din lumea naturală.
- O **imagine digitală** este o reprezentare numerică a unei imagini bidimensionale.

Grafică raster

- o **imagine** grafică **raster** este formată prin împărțirea zonei unei imagini într-o matrice dreptunghiulară de rânduri și coloane compuse din pixeli[1].
- un **pixel** este zonă pătrată (sau ocasional rectangulară) de culoare, reprezentând un singur punct din imagine [2].

0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	1	1	1	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	1	1	1	1	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	1	1	1	1	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0

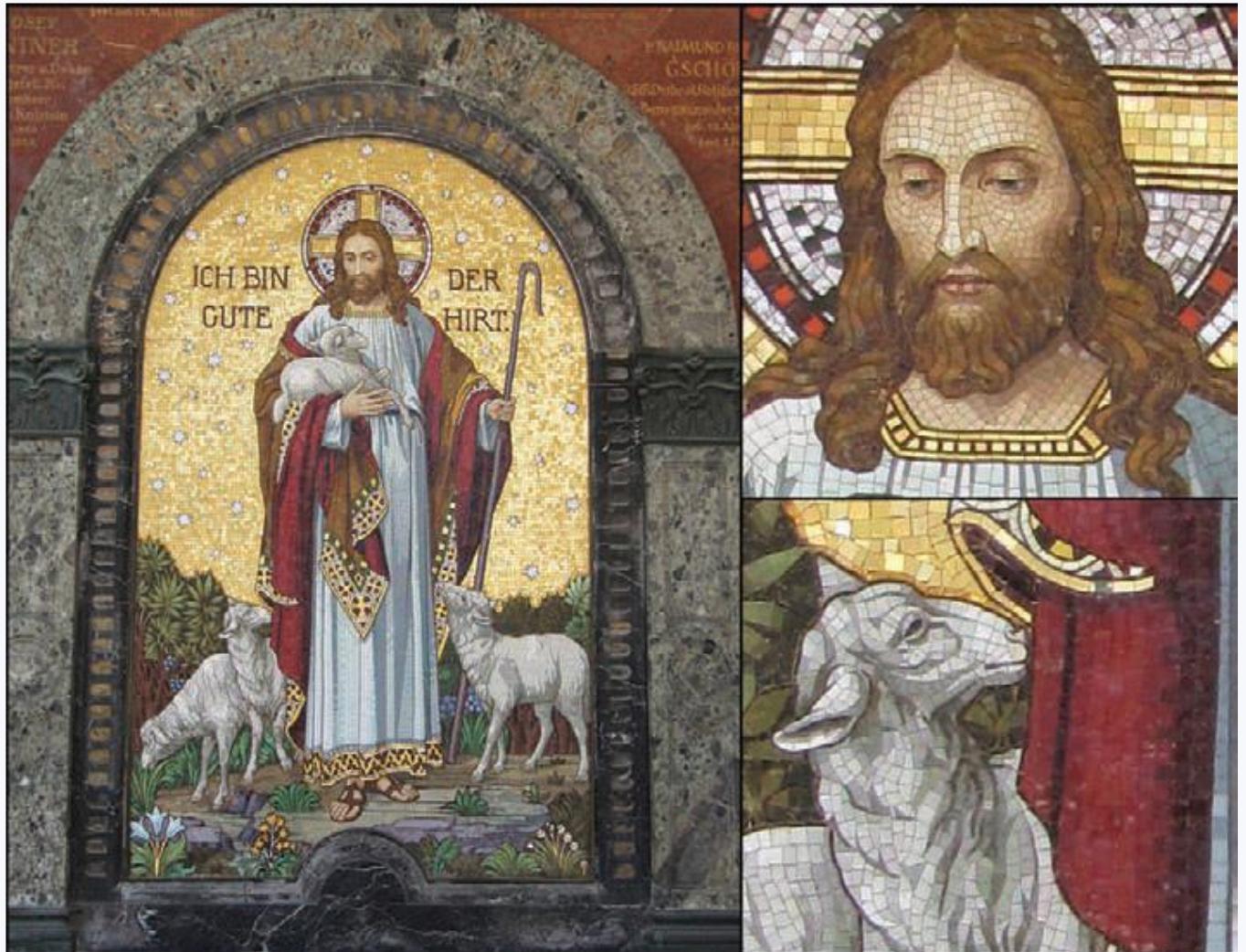


Grafică raster

- Toți pixelii dintr-o imagine raster au exact aceeași dimensiune. Fiecare specifică o singură culoare stocată utilizând 24 de biți.
- Numărul total de pixeli dintr-o imagine raster este fix. Pentru a mări o imagine raster este necesară adăugarea de pixeli suplimentari.
- Similar, anumiți pixeli trebuie eliberați le reducerea dimensiunilor unei imagini raster. Lățimea și înălțimea unei imagini raster sunt determinate de câte pixeli conține fiecare rând și coloană.

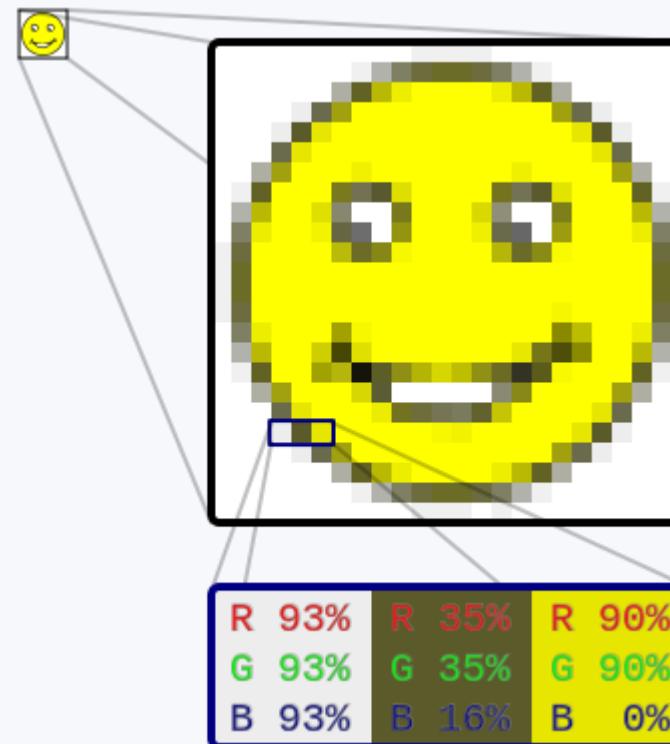
Mozaic

În cazul unui mozaic privit de la distanță, plăcile individuale care formează imaginea sunt abia perceptibile cu ochiul liber. Acestea devin însă vizibile dacă mozaicul este privit de aproape. Această tehnică, ce presupune utilizarea unor mici bucăți de material colorat a apărut cu aproximativ 3.000 î.Hr.



Mozaic din secolul al XIX-lea

Pixeli



Prin utilizarea facilității de zoom disponibilă în programele de grafică, pixelii individuali devin vizibili sub formă de pătrate.

Pixeli



Activitate practică:

1. Copiați imaginea de pe slide-ul anterior în editorul de imagini raster favorit (ex: MS Paint)
2. Măriți la maxim nivelul de zoom. Activați opțiunea gridlines, dacă este disponibilă.

Caracteristici

- **Rezoluție**
 - descrie calitatea unei imagini raster și se referă direct la dimensiunea și numărul de pixeli conținuți de imagine.
- **Dimensiunea în pixeli**
 - descrie dimensiunea unei imagini raster, exprimată ca numărul de pixeli de-a lungul axei x (lățime) și numărul de pixeli de-a lungul axei y (înăltime).
 - ex: 800 px * 600 px
- **Număr de pixeli**
 - Numărul de pixeli este numărul total de pixeli dintr-o matrice raster. Pentru a determina numărul de pixeli, se înmulțesc dimensiunile orizontale și verticale ale imaginii.
 - ex: o imagine cu dimensiunea de 30 * 18 pixeli conține 540 pixeli.

Caracteristici

- **Densitatea pixelilor**
 - Exprimăm densitatea pixelilor sau rezoluția unei imagini raster în pixeli pe inch (ppi) - aceasta reprezintă pixeli pe inch liniar (transversal sau în jos), nu inch pătrat..
 - Multe monitoare și televizare u o rezoluție de afișare de 72 sau 96 ppi.
 - Rezoluția determină dimensiunea maximă la care o imagine poate fi imprimată. Pentru a produce o imprimare de înaltă calitate de orice dimensiune, fotografiile digitale au nevoie de o densitate de pixeli de cel puțin 300 ppi - adică 90.000 pixeli într-un inch pătrat

Caracteristici

- **Adâncimea culorii (rezoluția culorii)**
 - măsură a numărului de culori diferite care pot fi reprezentate de un pixel individual. Numărul de biți alocate fiecărui pixel sau adâncimea de biți determină rezoluția culorii.
 - 8-bit (256 culori), 24 biti (16.8 milioane de culori)
 - Un desen folosind 16 culori distințe, de exemplu, ar necesita doar 4 biți pe pixel. Utilizarea unui cod cu o adâncime de biți mai mare produce un fișier mai mare, fără creșterea calității.

Avantaje și dezavantaje

- Avantaje:
 - excelent atunci când se creează imagini bogate și detaliate. Fiecare pixel dintr-o imagine raster poate avea o culoare diferită, prin urmare pot fi create imagini complexe cu orice fel de schimbări de culoare și variații.

Avantaje și dezavantaje

- Dezavantaje:
 - fișierele sunt adesea **destul de mari**, deoarece conțin toate informațiile pentru fiecare pixel al imaginii.
 - dacă este mărită, o imagine raster va arăta granulată și distorsionată, deoarece imaginile raster sunt create cu un număr finit de pixeli. La mărirea imaginii nu există informație pentru pixelii adăugați suplimentar. Software-ul de editare a fotografiilor va încerca să umple acești pixeli cât mai bine, cu toate acestea, imaginea rezultată va fi adesea neclară.
 - nu sunt furnizate informații cu privire la formele care alcătuiesc imaginea.

Avantaje și dezavantaje

Activitate practică:

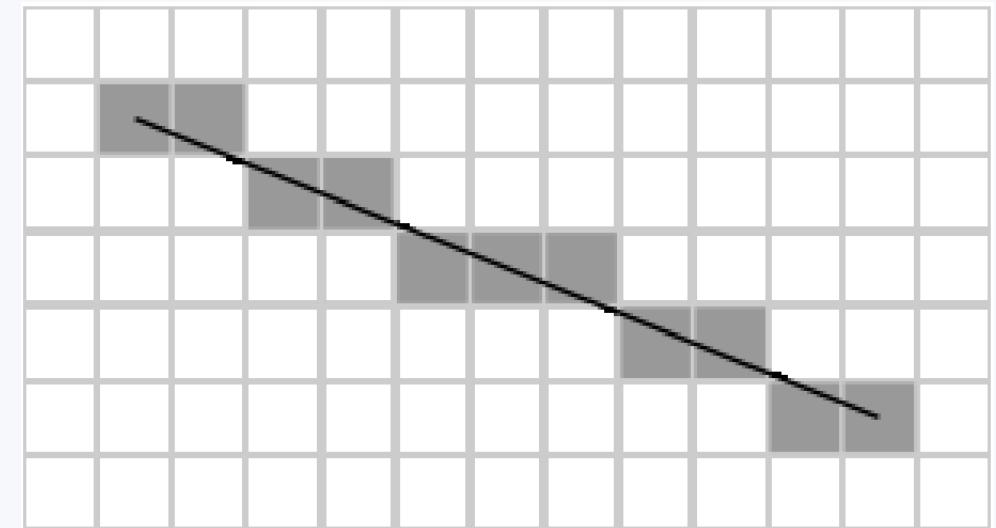
1. Salvați imaginile de mai jos și analizați modul în care reacționează la modificarea dimensiunilor.
2. Care dintre cele două imagini este de tip raster?



Desenare elemente grafice

Exemplu: Trasarea unei linii presupune colorarea celulelor matricei plecând de la o ecuație matematică

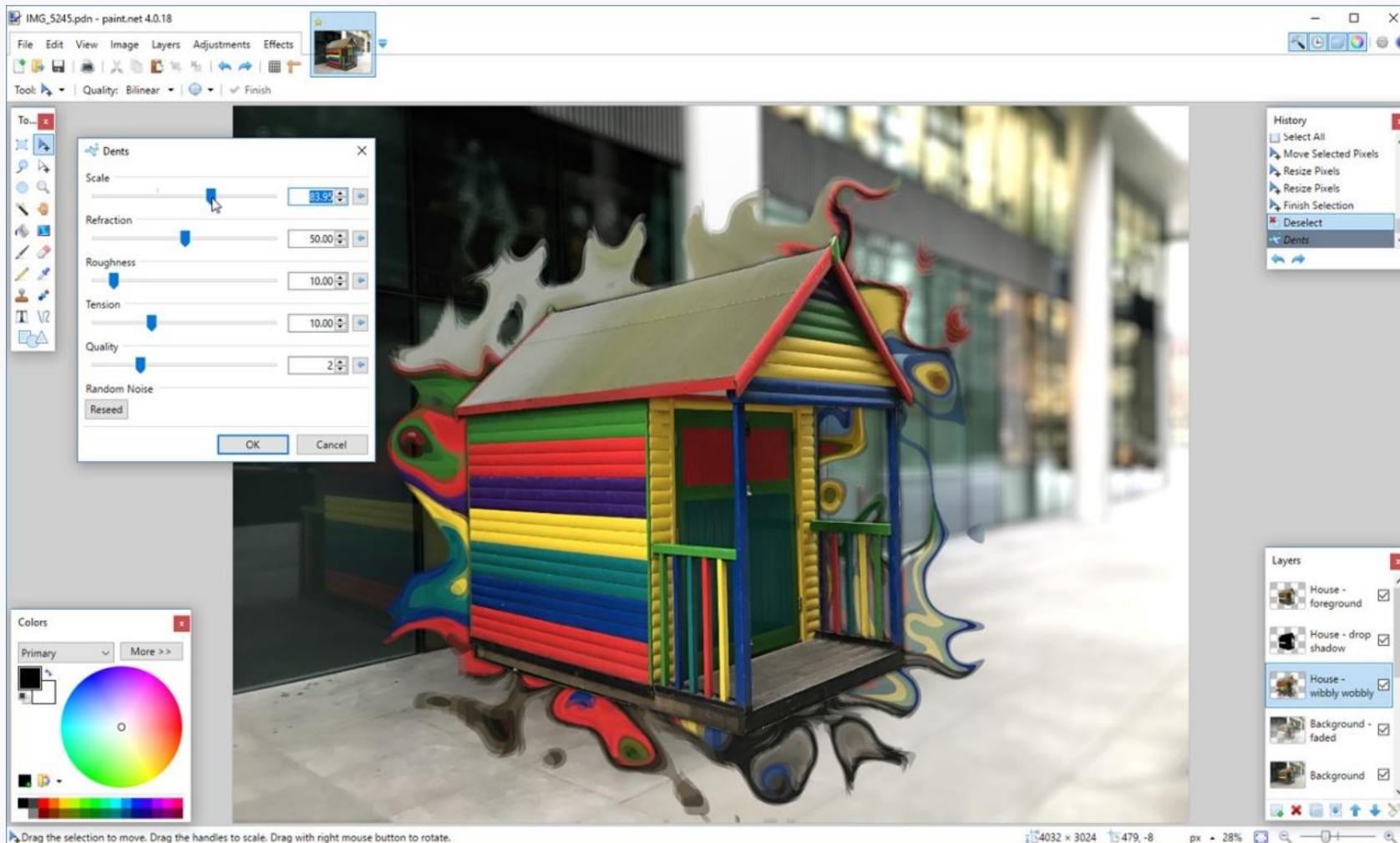
Exemplu: segment de dreaptă $(x_1, y_1) \rightarrow (x_2, y_2)$



Formate

- În timpul lucrului cu un fișier imagine, acesta va fi în general salvat într-un format care acceptă straturi (*.PSD, *TIFF) pentru a putea fi modificat cu ușurință.
- Pentru utilizarea imaginii într-un proiect multimedia starurile vor fi **combinate**, iar imaginea va fi comprimată.
- În cazul formatele de compresie cu pierderi există întotdeauna un compromis între calitatea imaginii și dimensiunea fișierului.
- Aplicații web: <https://www.photopea.com>, <https://pixlr.com/x/>

Grafică raster Formate



Utilizare straturi (en: layers) în Paint.NET

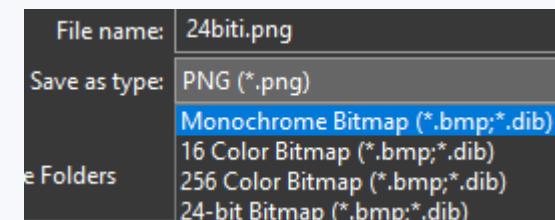
Formate

- **BMP** (Microsoft Windows Bitmap) - ***.bmp**
 - cunoscut și sub numele de fișier de imagine bitmap sau format de fișier bitmap independent de dispozitiv (DIB) sau pur și simplu bitmap
 - fie necomprimat, fie comprimat folosind RLE (format de compresie fără pierderi)
 - monocrom sau color (sunt acceptate diferite adâncimi de culoare)

Formate



1. Desenați în Microsoft Paint o formă geometrică simplă (ex: un dreptunghi roșu)
2. Comparați dimensiunile fișierelor obținute prin selectarea celor patru adâncimi de culoare disponibile.



Formate

- GIF (Graphics Interchange Format) - *.gif
 - maxim 256 de culori și transparentă (pixeli transparenti);
 - format de compresie fără pierderi.
- utilizat în mod obișnuit pentru sigle și alte imagini cu linii și blocuri de culoare solide.
- permite rularea de animații.

Formate



<https://dl.dropboxusercontent.com/u/70618257/HTML5Multimedia/giphy.gif>

Formate

- **JPEG** (Joint Photographic Experts Group) - ***.jpeg**
 - permite afișarea a 16.8 milioane de culori
 - nu suportă transparentă (nu permite definirea de pixeli transparenti).
 - format de compresie cu pierderi
 - este folosit cel mai adesea pentru fotografii

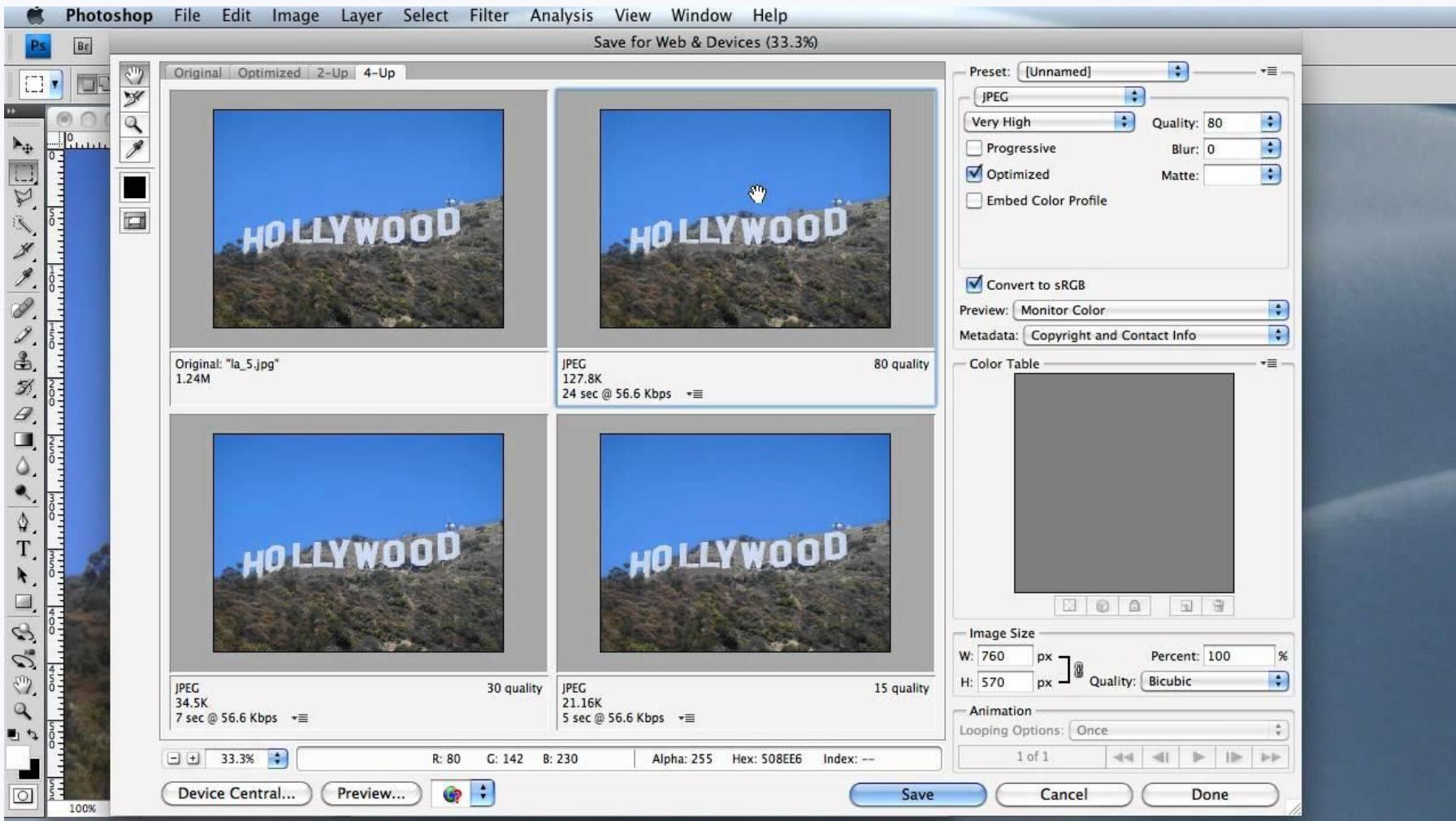
Formate



JPEG - o fotografie a unei pisici cu rata de compresie în scădere și, prin urmare, creșterea calității, de la stânga la dreapta[1].

Grafică raster

Formate

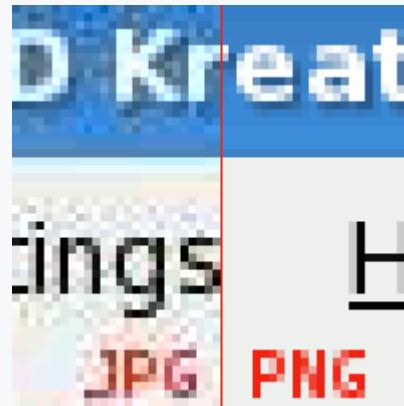


Photoshop – meniul Save for Web

Formate

- **PNG** (Portable Network Graphics) - *.png
 - cel mai utilizat format de compresie a imaginilor fără pierderi de pe Internet [1]
 - creat ca un înlocuitor îmbunătățit, nebrevetat, pentru formatul GIF [1]
 - permite afișarea a 16.8 million colors; suportă transparentă; permite utilizarea unui număr mai mic de culori pentru a reduce dimensiunea fișierului (PNG 8, sau PNG cu reprezentarea culorilor pe 8-biți) [2]
 - format de compresie fără pierderi[2]
 - utilizat pentru o gamă largă de imagini, inclusiv *favicons* (the small web page icons in browser tabs) [2]
 - fișierele PNG pot fi foarte mici, dar pentru fotografii cu multe culori, pot avea o dimensiune mai mare decât fișierele JPEG la o calitate comparabilă [2]

Formate



Imagine compusă care compară compresia cu pierderi în JPEG cu compresia fără pierderi în PNG: artefactele JPEG sunt ușor vizibile în fundal, unde imaginea PNG are o culoare solidă [1].

Formate

- WebP
 - acceptă compresia cu pierderi prin codificare predictivă bazată pe codecul video VP8 și compresia fără pierderi care folosește substituții pentru repetarea datelor.
 - codificarea predictivă folosește valorile din blocurile vecine de pixeli pentru a prezice valorile dintr-un bloc și apoi codifică doar diferența.
 - imaginile WebP cu pierderi sunt în medie cu 25–35% mai mici decât imaginile JPEG cu niveluri de compresie similare vizual. Imaginile WebP fără pierderi sunt de obicei cu 26% mai mici decât aceleasi imagini în format PNG.
 - permite redarea de animații
 - detalii: <https://developers.google.com/speed/webp> ,
https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types#WebP

Formate

- Other formats:
 - ICO - Icon Resource File;
 - DIB - Device Independent Bitmap;
 - PCX - PC PaintBrush File Format;
 - TIFF - Tag Image File Format.

Grafică raster

Imagini de dimensiuni mari

- Dubai: <http://gigapan.com/gigapans/48492>
- Size: 44.88 Gigapixels



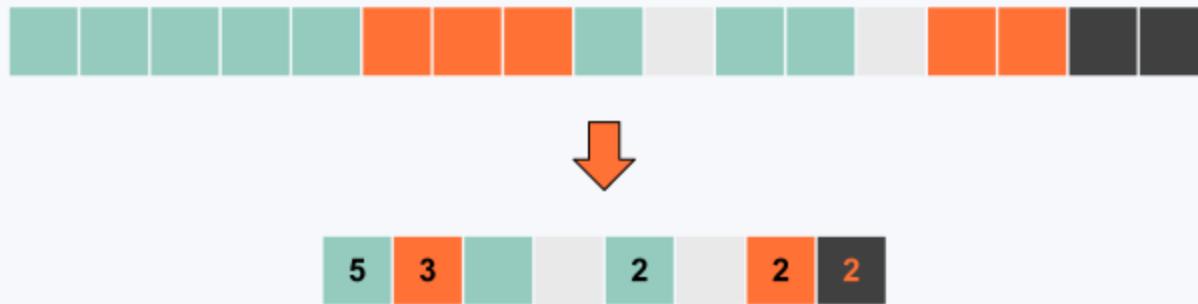
Compresie

- Algoritmi de compresie:
 - Run-length encoding (RLE)
 - Lempel–Ziv–Welch (LZW)
 - Huffman
 - JPEG

Compression - Run Length Encoding - RLE

- one of the simpler strategies to achieve **lossless** compression
- can be used to compress bitmapped image files (ex: *pcx format). Bitmapped images can easily become very large because each pixel is represented with a series of bits that provide information about its color.
- RLE generates a code to “flag” the beginning of a line of pixels of the same color. That color information is then recorded just once for each pixel. In effect, RLE tells the computer to repeat a color for a given number of adjacent pixels rather than repeating the same information for each pixel over and over. The RLE compressed file will be smaller, but it will retain all the original image data—it is “lossless.”

Compression - Run Length Encoding - RLE



Compression - Lempel–Ziv–Welch (LZW)

- lossless data compression algorithm
- used in the GIF image format
- **Encoding**
 - Initialize the dictionary to contain all strings of length one.
 - Find the longest string W in the dictionary that matches the current input.
 - Emit the dictionary index for W to output and remove W from the input.
 - Add W followed by the next symbol in the input to the dictionary.
 - Go to Step 2.

JPEG Compression

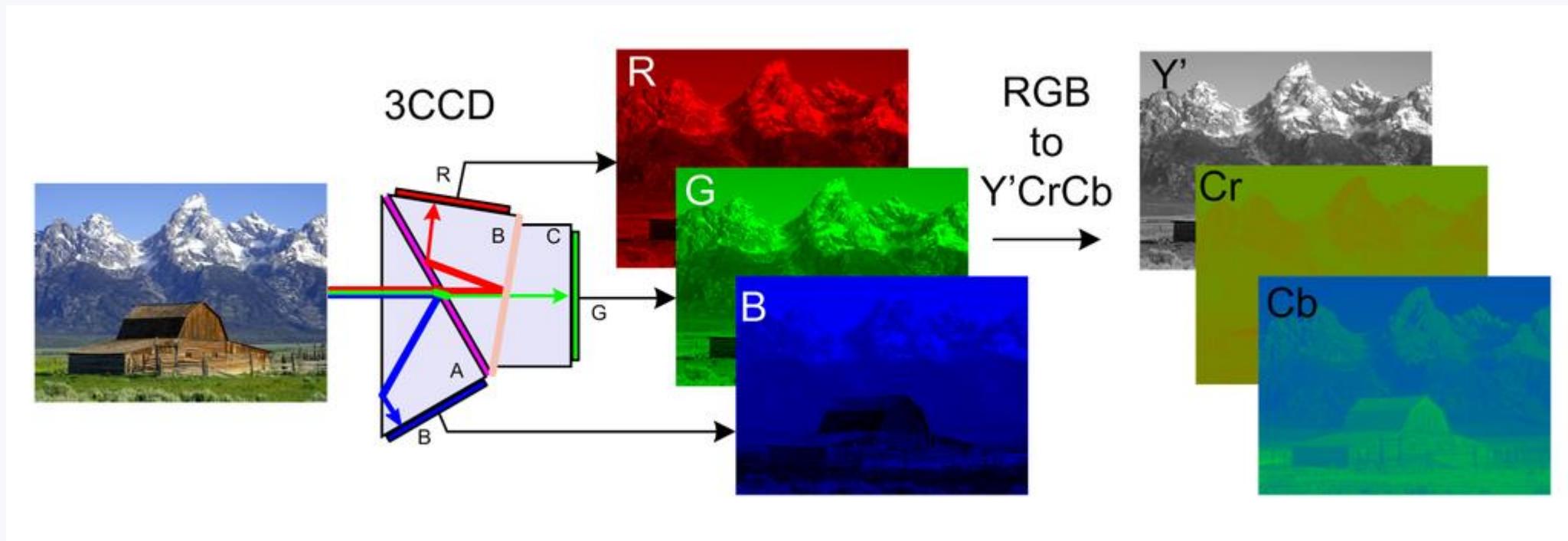
- lossy data compression algorithm

- Steps

1. color space transformation
2. downsampling
3. block splitting
4. transformation
5. quantization
6. encoding

JPEG Compression - Step 1 - Color space transformation

- The representation of the colors in the image is converted from RGB to Y'CrCb, consisting of one luma component (Y'), representing brightness, and two chroma components, (CB and CR), representing color. This step is sometimes skipped.



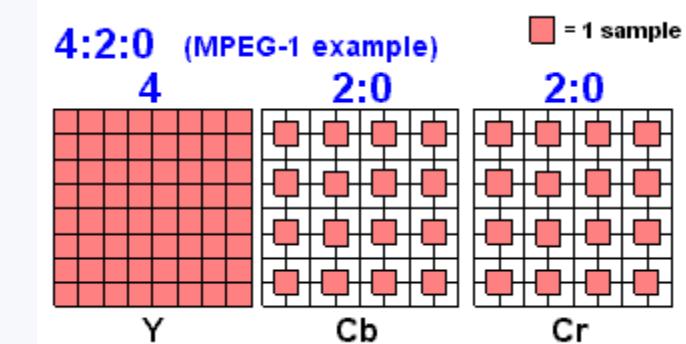
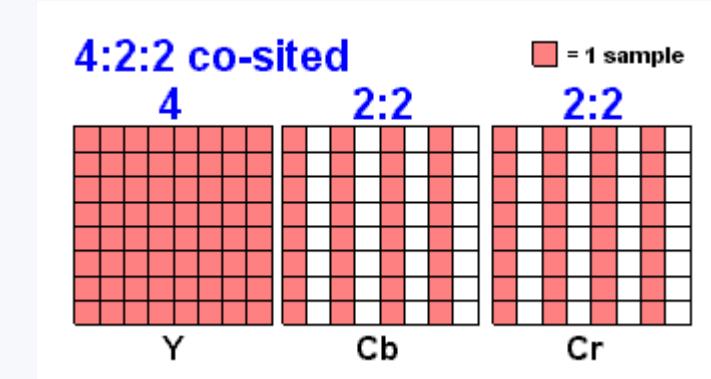
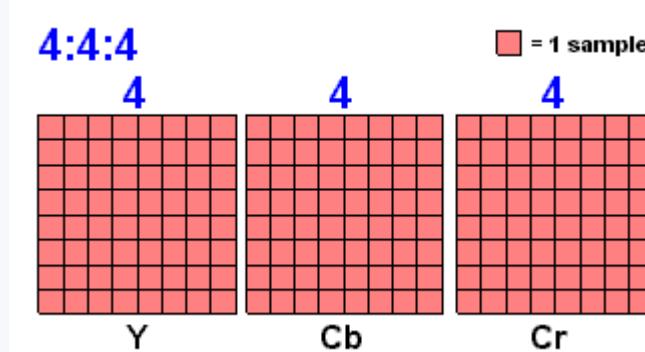
JPEG Compression - Step 1 - Color space transformation

- The $Y'C_BC_R$ color space conversion allows greater compression without a significant effect on perceptual image quality (or greater perceptual image quality for the same compression).
- The compression is more efficient because the brightness information, which is more important to the eventual perceptual quality of the image, is confined to a single channel. This more closely corresponds to the perception of color in the human visual system.

JPEG Compression – Step 2 - Downsampling

- Due to the densities of color- and brightness-sensitive receptors in the human eye, humans can see considerably more fine detail in the brightness of an image (the Y' component) than in the hue and color saturation of an image (the C_b and C_r components). Using this knowledge, encoders can be designed to compress images more efficiently.
- The transformation into the Y'C_BC_R color model enables the next usual step, which is to reduce the spatial resolution of the C_b and C_r components.

JPEG Compression – Step 2 - Downsampling



JPEG Compression – Step 2 - Downsampling

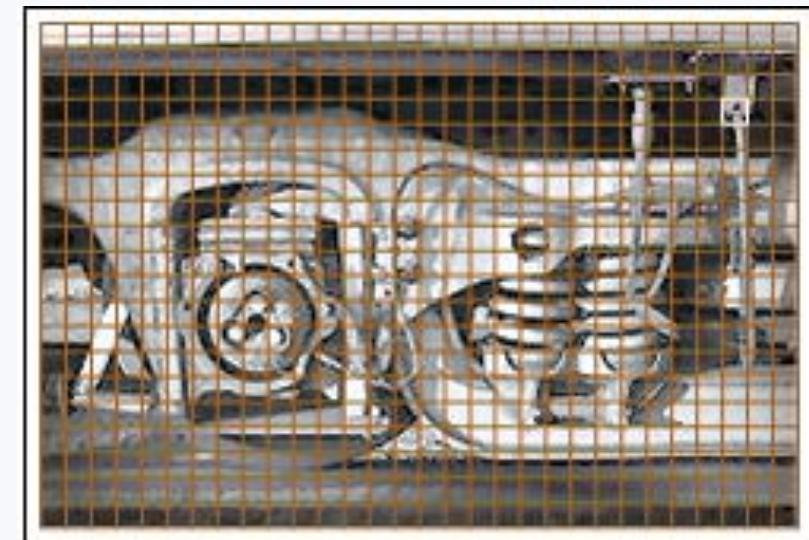
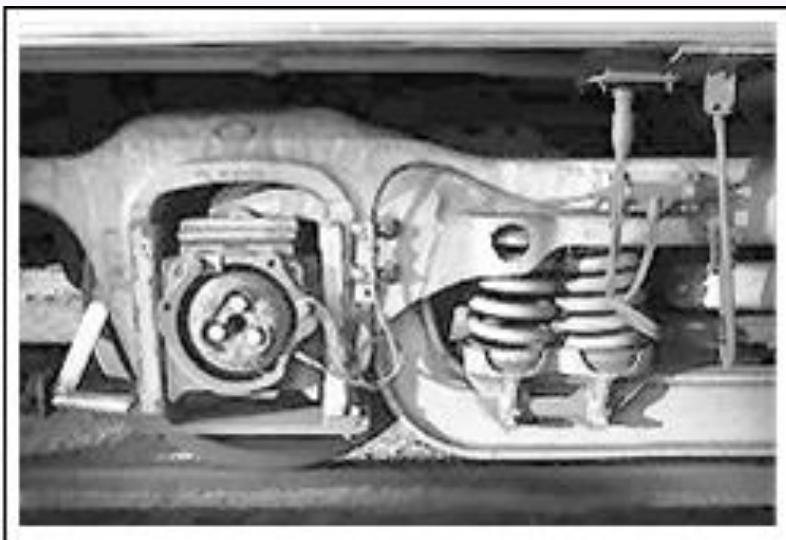
- The ratios at which the downsampling is ordinarily done for JPEG images are:
 - 4:4:4 - no downsampling,
 - 4:2:2 - reduction by a factor of 2 in the horizontal direction
 - 4:2:0 - reduction by a factor of 2 in both the horizontal and vertical directions (most common).
- For the rest of the compression process, Y', Cb and Cr are processed separately and in a very similar manner.

JPEG Compression – Step 3 - Block splitting

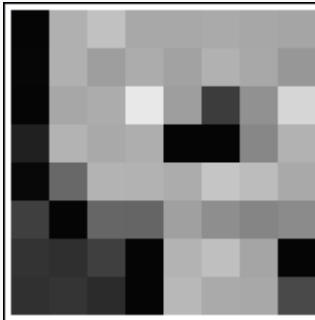
- Each channel must be split into 8×8 blocks
- Depending on chroma subsampling, this yields Minimum Coded Unit (MCU) blocks of size 8×8 (4:4:4 – no subsampling), 16×8 (4:2:2), or most commonly 16×16 (4:2:0).

JPEG Compression – Step 4 - Discrete cosine transform

- Each channel must be split into blocks of size 8 x 8 pixels
- If we have chosen an image whose dimensions are $160 \times 240 = 8*20 \times 8*30$. So this step creates $20 \times 30 = 600$ blocks.



JPEG Compression – Step 4 - Discrete cosine transform



5	176	193	168	168	170	167	165
6	176	158	172	162	177	168	151
5	167	172	232	158	61	145	214
33	179	169	174	5	5	135	178
8	104	180	178	172	197	188	169
63	5	102	101	160	142	133	139
51	47	63	5	180	191	165	5
49	53	43	5	184	170	168	74

JPEG Compression – Step 4 - Discrete cosine transform

- Before computing the DCT of the 8×8 block, its values are shifted from a positive range to one centered on zero.
- We subtract 127 from each pixel intensity in each block. This step centers the intensities about the value 0 and it is done to simplify the mathematics of the transformation and quantization steps.

JPEG Compression – Step 4 - Discrete cosine transform

5	176	193	168	168	170	167	165
6	176	158	172	162	177	168	151
5	167	172	232	158	61	145	214
33	179	169	174	5	5	135	178
8	104	180	178	172	197	188	169
63	5	102	101	160	142	133	139
51	47	63	5	180	191	165	5
49	53	43	5	184	170	168	74

-122	49	66	41	41	43	40	38
-121	49	31	45	35	50	41	24
-122	40	45	105	31	-66	18	87
-94	52	42	47	-122	-122	8	51
-119	-23	53	51	45	70	61	42
-64	-122	-25	-26	33	15	6	12
-76	-80	-64	-122	53	64	38	-122
-78	-74	-84	-122	57	43	41	-53

JPEG Compression – Step 4 - Discrete cosine transform

- The JPEG Image Compression Standard relies on the *Discrete Cosine Transformation (DCT)* to transform the image. The DCT is applied to each 8×8 block.
- The DCT is a product $C = U B U^T$ where B is an 8×8 block from the preprocessed image and U is a special 8×8 matrix.
- Loosely speaking, the DCT tends to push most of the high intensity information (larger values) in the 8×8 block to the upper left-hand of C with the remaining values in C taking on relatively small values.

JPEG Compression – Step 4 - Discrete cosine transform

-27.500	-213.468	-149.608	-95.281	-103.750	-46.946	-58.717	27.226
168.229	51.611	-21.544	-239.520	-8.238	-24.495	-52.657	-96.621
-27.198	-31.236	-32.278	173.389	-51.141	-56.942	4.002	49.143
30.184	-43.070	-50.473	67.134	-14.115	11.139	71.010	18.039
19.500	8.460	33.589	-53.113	-36.750	2.918	-5.795	-18.387
-70.593	66.878	47.441	-32.614	-8.195	18.132	-22.994	6.631
12.078	-19.127	6.252	-55.157	85.586	-0.603	8.028	11.212
71.152	-38.373	-75.924	29.294	-16.451	-23.436	-4.213	15.624

JPEG Compression – Step 5 - Quantization

- elements near zero will converted to zero and other elements will be shrunk so that their values are closer to zero. All quantized values will then be rounded to integers.
- quantization is performed in order to obtain integer values and to convert a large number of the values to 0. The Huffman coding algorithm will be much more effective with quantized data and the hope is that when we view the compressed image, we haven't given up too much resolution.

JPEG Compression – Step 5 - Quantization

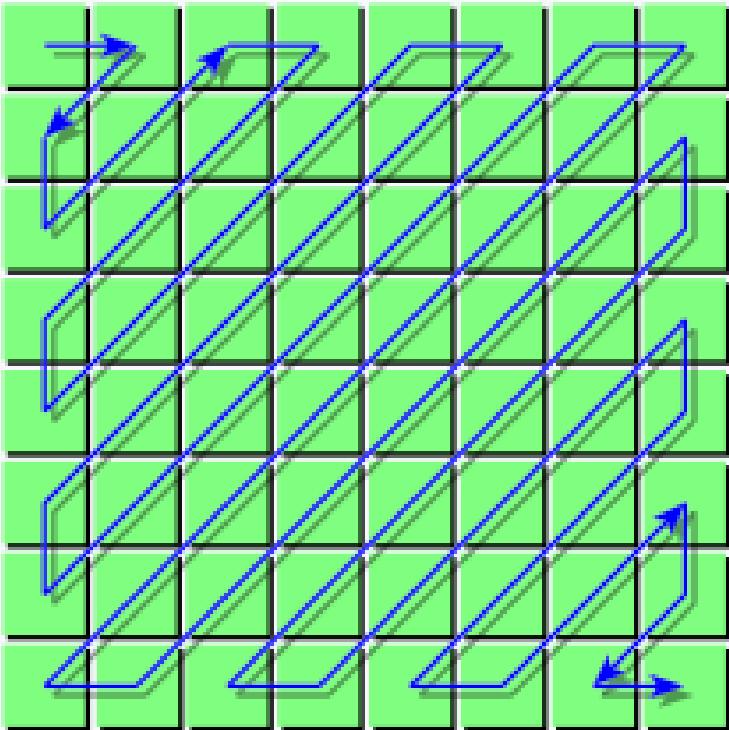
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Note that the values are largest in the lower right corner of Z. These divides should produce values close to zero so that the rounding function will convert the quotient to zero

JPEG Compression – Step 5 - Quantization

-2	-19	-15	-6	-4	-1	-1	0
14	4	-2	-13	0	0	-1	-2
-2	-2	-2	7	-1	-1	0	1
2	-3	-2	2	0	0	1	0
1	0	1	-1	-1	0	0	0
-3	2	1	-1	0	0	0	0
0	0	0	-1	1	0	0	0
1	0	-1	0	0	0	0	0

JPEG Compression – Step 6 - Encoding



It involves arranging the image components in a "zigzag" order employing run-length encoding (RLE) algorithm that groups similar frequencies together, inserting length coding zeros, and then using **Huffman** coding on what is left.

JPEG Compression – Step 5 - Quantization

- quantization makes the JPEG algorithm an example of *lossy compression*:
 - the DCT step is completely invertible - that is, we applied the DCT to each block B by computing $C = U B U^T$. It turns out we can recover B by the computation $B = U^T C U$.
 - converting small values to 0 and rounding all quantized values are not reversable steps. The ability to recover the original image is lost.

HTML5 Image Element

- afişarea imaginilor în HTML :

```
<img alt="" src="" title="">
```

- tipuri de imagini raster suportate: JPEG, GIF, PNG, WEBP
- Element: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img>
- API: <https://developer.mozilla.org/en-US/docs/Web/API/HTMLImageElement>

HTML5 Image Element

- Proprietăți:
 - width, height: dimensiunea obiectului în fereastră;
 - naturalWidth, naturalHeight: dimensiunea matricei raster;
 - src: URL-ul imaginii sursă; modificarea determină începutul procesului de încărcare (asincron).
- Evenimente:
 - load: declanșat când o resursă și resursele dependente ale acesteia s-au terminat de încărcat.

HTML5 Image Element

```
//jQuery
let image = $("<img>")
    .load(function () { $("body").append($(this)); })
    .attr("src", "media/Penguins.jpg");
```

```
//Vanilla JavaScript
let image = document.createElement('img');
image.addEventListener('load', function(e){
    document.body.appendChild(e.target)
});
image.setAttribute('src', "mask.png");
```

HTML5 Image Element



Rescrieți codul din slide-ul anterior utilizând "promise".

HTML5 Canvas Element

- Poate fi folosit pentru a desena grafice, a face compozitii foto sau chiar a efectua animatii.
- Declararea unui element de tip canvas :

```
<canvas id="test" style="width:250px; height:150px">  
    An alternative text describing what your canvas displays.  
</canvas>
```

- API: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas>
- DOM interface: HTMLCanvasElement

HTML5 Canvas Element - Metode

- `getContext(contextType, contextAttributes);` - returnează contextul grafic de desenare al canvas-ului sau null, dacă contextul solicitat nu este suportat;

```
let canvas = document.getElementById('test'); // sau let canvas = $('test')[0];
let w = canvas.width, h = canvas.height;
let ctx = canvas.getContext('2d');
```

- `contextType`: specifică tipul de context grafic utilizat pentru desenarea pe canvas.
- `contextAttributes`: setări suplimentare
- docs: developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/getContext

HTML5 Canvas Element - Metode

- valori posibile contextType:
 - "2d" - determină returnarea unui obiect de tipul `CanvasRenderingContext2D` reprezentând un context de desenare bidimensională.
 - "webgl" / "webgl2" - determină returnarea unui obiect de tipul `WebGLRenderingContext` reprezentând un context de desenare tridimensională. Acest tip de context grafic este disponibil doar în browser-ele care implementează WebGL version 1 (OpenGL ES 2.0) / 2 (OpenGL ES 3.0).
 - "bitmaprenderer" - determină returnarea unui obiect de tip `ImageBitmapRenderingContext` care oferă doar funcționalitatea necesară pentru a înlocui conținutul canvas-ului cu un `ImageBitmap`.

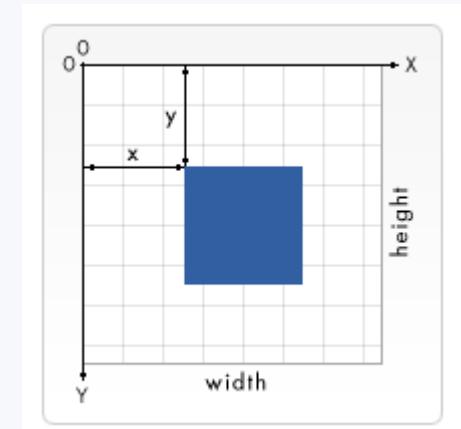
HTML5 Canvas Element - Metode



1. Verificați tipul de element HTML utilizat de maps.google.com pentru afișarea hărții.
2. Care credeți că este tipul de context grafic utilizat?

HTML5 Canvas Element - Interfața CanvasRenderingContext2D

- Utilizată pentru desenarea de formelor geometrice, imaginilor, textelor și a altor elemente pe <canvas>.
- API: developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D
- Desenarea de dreptunghiuri:
 - `clearRect(x, y, width, height)` - ștergere suprafață
 - `fillRect(x, y, width, height)` - umplere suprafață
 - `strokeRect(x, y, width, height)` - desenare contur



HTML5 Canvas Element - Interfața CanvasRenderingContext2D

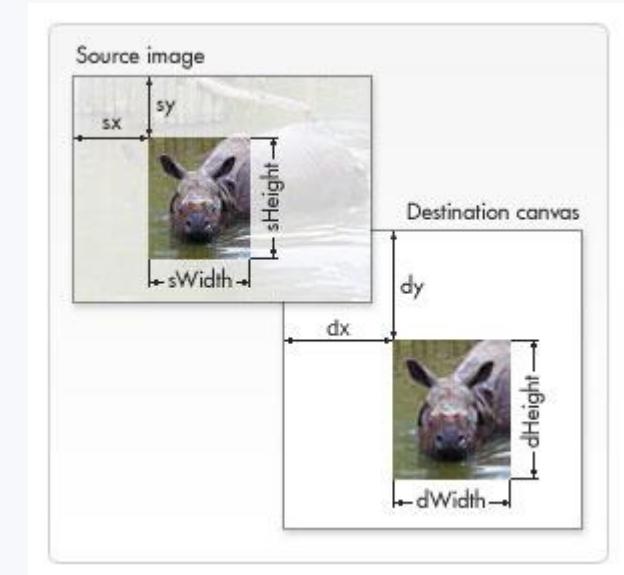
- Desenarea textului:
 - `fillText(text, x, y [, maxWidth])` – desenare text (cu umplere);
 - `strokeText(text, x, y [, maxWidth])` – desenare text (doar contur);
 - `measureText(text)` - returns a `TextMetrics` object (include o proprietate `width`).
- Desenare folosind căi:
 - [documentație](#)
- Stiluri pentru linii:
 - [documentație](#)

HTML5 Canvas Element - Interfața CanvasRenderingContext2D

- Stiluri pentru text:
 - check [documentation](#)
- Stiluri pentru umplere și contur (en: Fill and stroke styles):
 - check [documentation](#)
- Gradient și pattern:
 - check [documentation](#)
- Umbre:
 - check [documentation](#)

HTML5 Canvas Element - Interfața CanvasRenderingContext2D

- Desenarea imaginilor:
 - fără scalare
 - `void ctx.drawImage(image, dx, dy);`
 - cu scalare
 - `void ctx.drawImage(image, dx, dy, dWidth, dHeight);`
 - `void ctx.drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight);`
 - API: [Mozilla Developer Network](#)



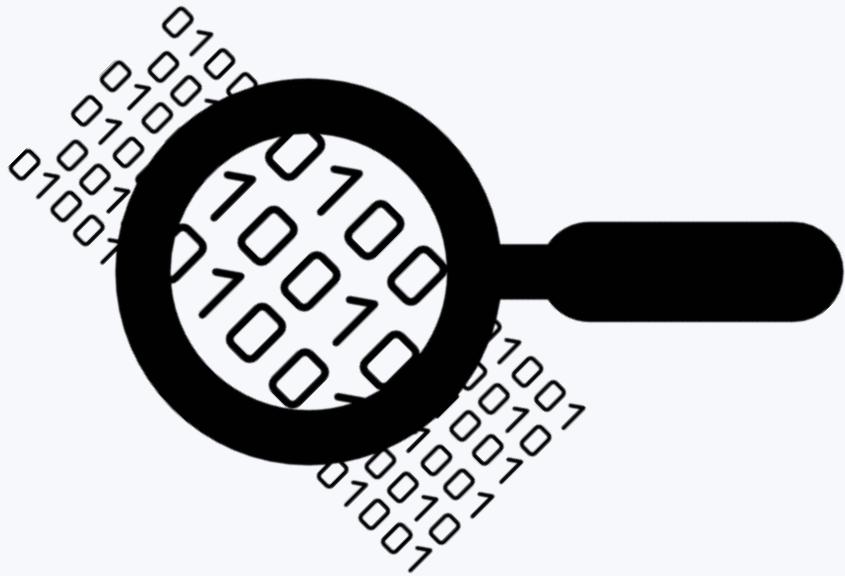
Salvarea și restabilirea setărilor

- **CanvasRenderingContext2D.save()**
 - salvează setările de desenare prin adăugarea acestora la o stivă
- **CanvasRenderingContext2D.restore()**
 - modifică setările de desenare prin extragerea primei intrări din stiva utilizată pentru salvarea setărilor

Salvarea și restabilirea setărilor

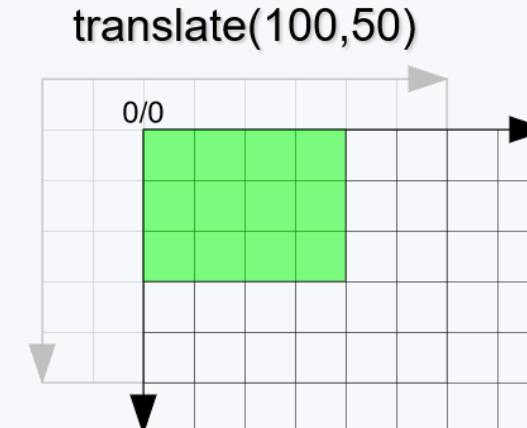
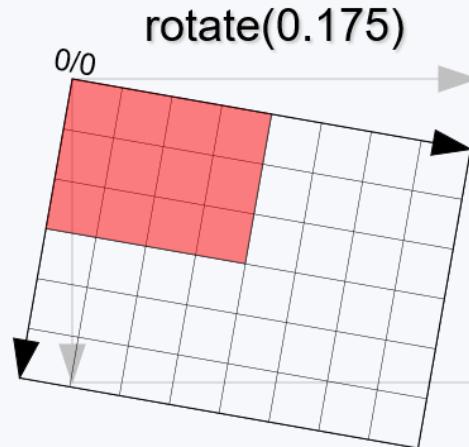
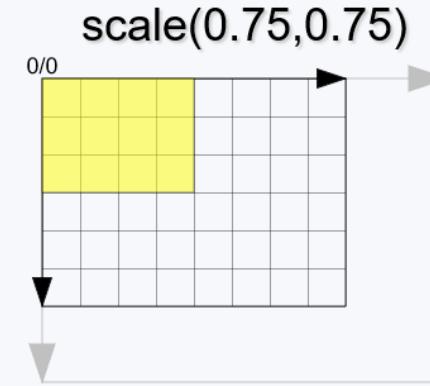
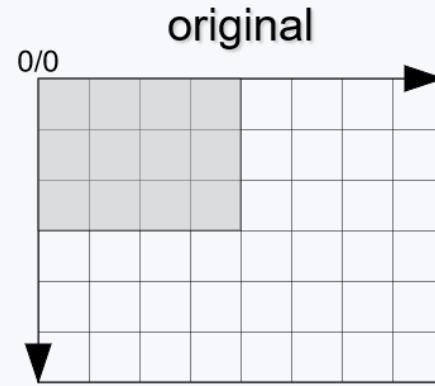
- Setări salvate:
 - matricea curentă de transformare.
 - zona curentă de desenare (en: clipping region).
 - setările pentru linii punctate (en: dash list).
 - atributele: strokeStyle, fillStyle, globalAlpha, lineWidth, lineCap, lineJoin, miterLimit, lineDashOffset, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor, globalCompositeOperation, font, textAlign, textBaseline, direction, imageSmoothingEnabled.

Salvarea și restabilirea setărilor



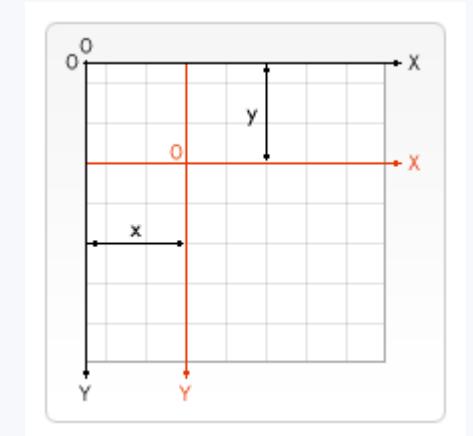
- save și restore

HTML5 Canvas Element - Transformări



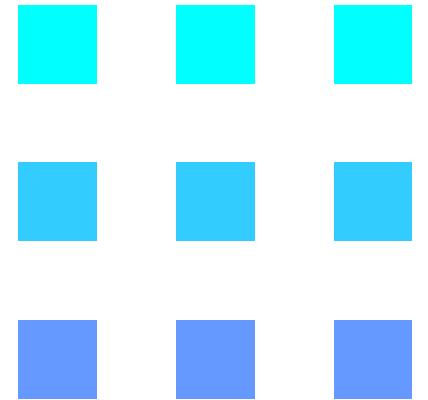
HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Translație
 - `translate(x, y)`
 - modifică originea.
 - x indică deplasarea pe orizontală și y indică deplasarea pe verticală.
 - API: [Mozilla Developer Network](#)



HTML5 Canvas Element - CanvasRenderingContext2D Interface

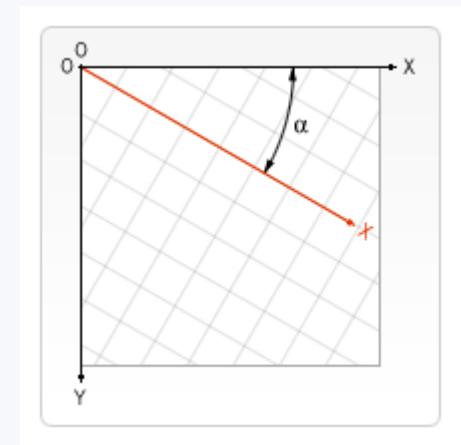
```
function draw() {
  let ctx = document.getElementById('canvas').getContext('2d');
  for (let i=0;i<3;i++) {
    for (let j=0;j<3;j++) {
      ctx.save();
      ctx.fillStyle = 'rgb('+(51*i)+','+(255-51*i)+',255)';
      ctx.translate(10+j*50,10+i*50);
      ctx.fillRect(0,0,25,25);
      ctx.restore();
    }
  }
}
```



- Fără apelarea metodei translate() toate dreptunghiurile ar fi desenate (0,0). Utilizând metoda translate() nu mai este necesară modificarea parametrilor metodei fillRect().
- JSFiddle: <https://jsfiddle.net/liviucotfas/9tfdegn7/>

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Rotație
 - `rotate(unghi)`
 - rotește sistemul de coordonate cu unghiul specificat (în radiani)
 - rotația se efectuează față de originea canvas-ului, cu excepția cazului în care aceasta a fost modificată cu ajutorul metodei `translate`
 - Notă: Unghiul este exprimat în radiani, nu în grade.
Conversie: radians = $(\text{Math.PI}/180) * \text{degrees}$.
 - API: [Mozilla Developer Network](#)



HTML5 Canvas Element - CanvasRenderingContext2D Interface

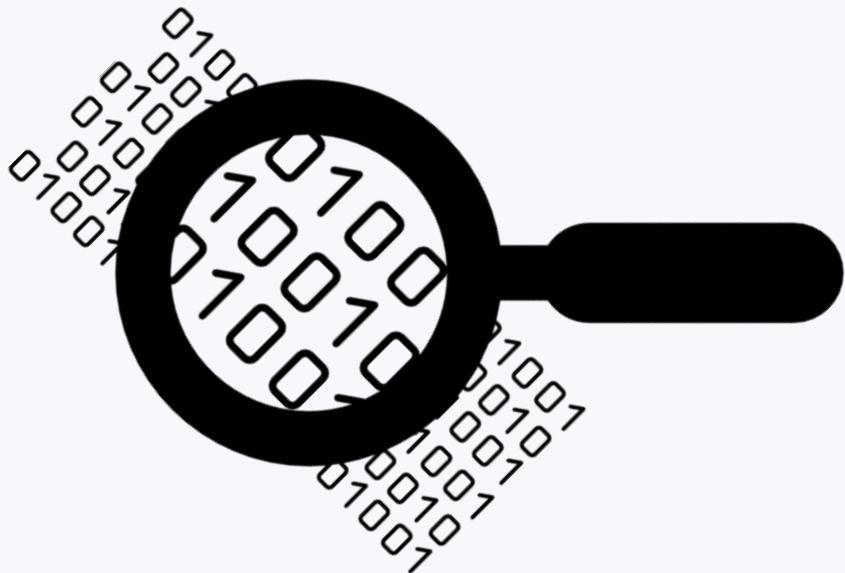


- dreptunghiul 1: rotit față de originea elementului de tip <canvas>
- dreptunghiul 2: rotit față de centrul dreptunghiului cu ajutorul metodei **translate()**.
- JSFiddle: <https://jsfiddle.net/liviucotfas/2yf241q1/>

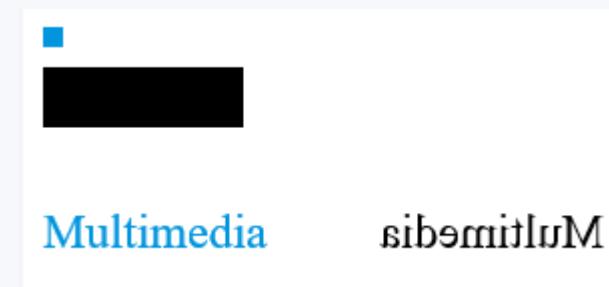
HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Scalare
 - `scale(x, y)`
 - Modifică dimensiunea unității de desenare pe canvas pe orizontală sau pe verticală.
 - Ambii parametri sunt numere reale. Valorile mai mici de 1,0 reduc dimensiunea unității de desenare, iar valorile peste 1,0 măresc dimensiunea unității de desenare. Valorile de 1,0 lasă unitatea de aceeași dimensiune.
 - Utilizând valori negative putem desena în “oglindă”.
 - API: [Mozilla Developer Network](#)

HTML5 Canvas Element - CanvasRenderingContext2D Interface



- rectangle: scaled
- text: mirrored.
- JSFiddle: <https://jsfiddle.net/liviucotfas/Lyycq7gv/>



Game Development



Question 1: SVG or Canvas?

Question 2: Context de desenare utilizat?
(webgl, webgl2, 2d)?

<https://techcrunch.com/2020/10/29/google-brings-halloween-to-life-using-augmented-reality/>

<https://www.google.com/doodles/halloween-2020>

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Combinare transformări
 - browser-ul utilizează o matrice de transformare

```
context.scale(-.5, 1);
context.rotate(45 * Math.PI / 180);
context.translate(40, 10);
```

- translația, rotația și scalarea modifică valorile din această matrice

$$\begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Transformări
 - permit modificări directe ale matricii de transformare
 - `transform(m11, m12, m21, m22, dx, dy)`
 - înmulțește matricea de transformare curentă cu matricea descrisă de argumentele sale.
 - API: [Mozilla Developer Network](#)

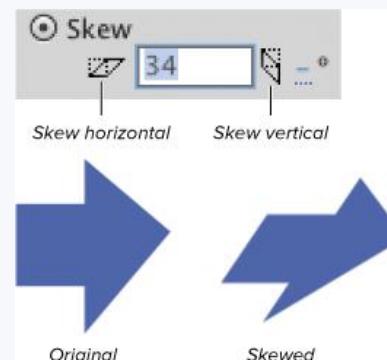
$$\begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- m11 - Horizontal scaling.
- m12 - Horizontal skewing.
- m21 - Vertical skewing.
- m22 - Vertical scaling.
- dx - Horizontal moving.
- dy - Vertical moving.

transform(m11, m12, m21, m22, dx, dy)

$$\begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

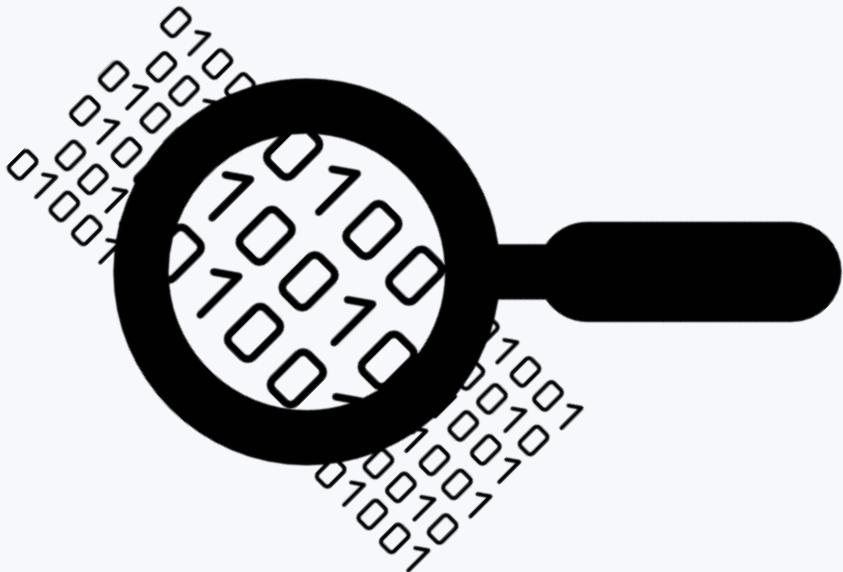


HTML5 Canvas Element - CanvasRenderingContext2D Interface

- ex: transformare coordonate

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

HTML5 Canvas Element - CanvasRenderingContext2D Interface



1. Utilizați metoda “transform” pentru a scrie codul echivalent cu “context.translate(50, 50);”
2. Utilizați metoda “transform” pentru a scrie codul echivalent cu “context.scale(1, -1);”

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- `resetTransform()`
 - reinițializează matricea de transformare curentă. Este echivalentă cu apelul:
`ctx.setTransform(1, 0, 0, 1, 0, 0);`
- `setTransform(m11, m12, m21, m22, dx, dy)`
 - reinițializează matricea de transformare curentă și apoi apelează metoda `transform()` cu argumentele specificate.
 - echivalentă cu apelarea metodelor "`resetTransform()`" și "`transform()`"

HTML5 Canvas Element - ImageData Interface

- Interfața ImageData oferă acces la pixelii care compun o anumită zonă din elementul de tip <canvas>.
- API: [Mozilla Developer Network](#)
- Putem obține o instanță de tip ImageData:
 - new ImageData();
 - CanvasRenderingContext2D.createImageData();
 - CanvasRenderingContext2D.getImageData();

HTML5 Canvas Element - ImageData Interface

- `new ImageData():` utilă în context webworker
- `CanvasRenderingContext2D.createImageData(sw, sh):` creează un obiect ImageData nou, gol, cu dimensiunile specificate. Toți pixelii din noul obiect sunt negri transparenti.
- `CanvasRenderingContext2D.createImageData(imageData):` creează un obiect ImageData nou, gol, cu dimensiunile obiectului imageData primit ca parametru. Toți pixelii din noul obiect sunt negri transparenti.
- `CanvasRenderingContext2D.getImageData(sx, sy, sw, sh):` extrage o porțiune din imagine ca obiect ImageData

HTML5 Canvas Element - ImageData Interface

- `CanvasRenderingContext2D.putImageData(imagedata, dx, dy)`: aplică datele pe imaginea afișată în Canvas

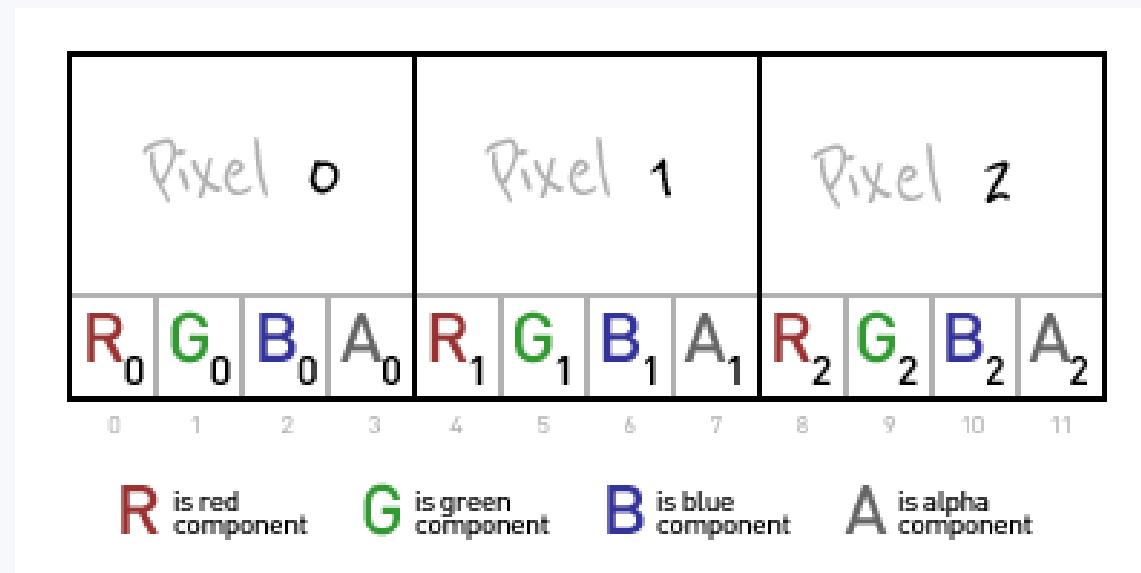
HTML5 Canvas Element - ImageData Interface



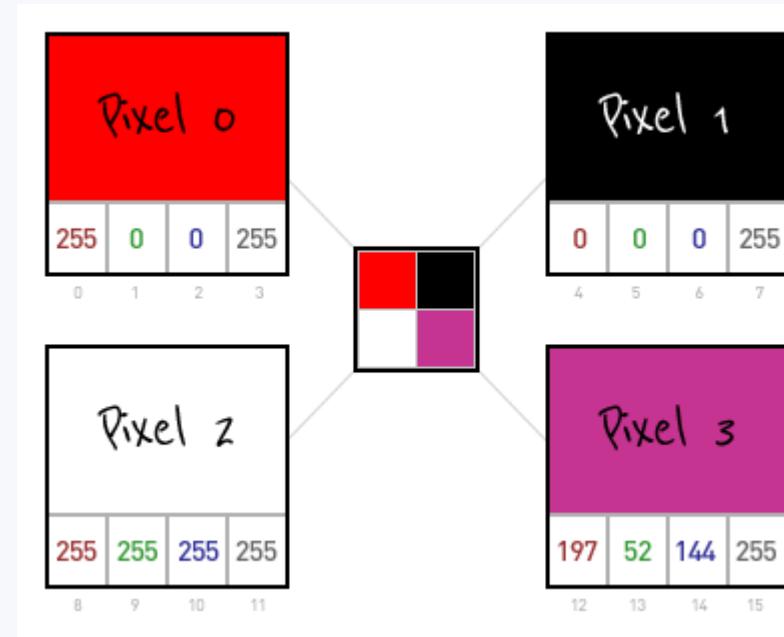
- Utilizați metodele `getImageData` și `putImageData` pentru a copia o porțiune dintr-un canvas.

HTML5 Canvas Element - ImageData Interface

- Proprietăți
 - `ImageData.data` : `Uint8ClampedArray` = vector care conține informațiile cu privire la culoarea pixelilor în ordinea RGBA, cu valori între 0 și 255.



HTML5 Canvas Element - ImageData Interface



```
[255,0,0,255, 0,0,0,255, 255,255,255,255, 197,52,144,255]
```

HTML5 Canvas Element - ImageData Interface



1. Creați o imagine similară cu cea de pe slide-ul anterior.
2. Verificați rezultatul întors de metoda `getImageData`.

HTML5 Canvas Element - ImageData Interface

- `ImageData.height` - unsigned long reprezentând înălțimea reală, în pixeli, a `ImageData`.
- `ImageData.width` - unsigned long reprezentând lățimea reală, în pixeli, a `ImageData`.

HTML5 Canvas Element - ImageData Interface

- Parcugerea tuturor pixelilor din canvas:

```
const imageData = context.getImageData(0, 0, canvas.width, canvas.height);
const data = imageData.data;

for (let i = 0; i < data.length; i+=4) {
    const red = imageData.data[i]; // [0..255]
    const green = imageData.data[i+1]; // [0..255]
    const blue = imageData.data[i+2]; // [0..255]
    const transparency = imageData.data[i+3]; // [0..255]
}
```

HTML5 Canvas Element - ImageData Interface

- Parcugerea tuturor pixelilor din canvas:

```
const imageData = context.getImageData(0, 0, canvas.width, canvas.height);

for (let y = 0; y < canvas.height; y++) {
    for (let x = 0; x < canvas.width; x++) {
        const i = (y * canvas.width * 4) + x * 4;

        const red = imageData.data[i]; // [0..255]
        const green = imageData.data[i+1]; // [0..255]
        const blue = imageData.data[i+2]; // [0..255]
        const transparency = imageData.data[i+3]; // [0..255]
    }
}
```

HTML5 Canvas Element - ImageData Interface



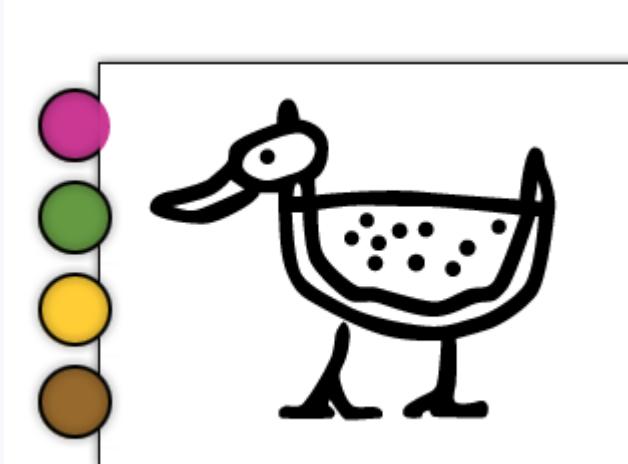
- Pornind de la codul prezentat în slide-ul anterior scrieți o funcție care transformă în tonuri de gri doar jumătatea stângă a unei imagini.

HTML5 Canvas Element

- Examples:
 - Color picker: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Pixel manipulation with canvas#A color picker
 - Zoom: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Pixel manipulation with canvas#A color picker

HTML5 Canvas Element

- Exemple:
 - <http://www.williammalone.com/articles/html5-canvas-javascript-paint-bucket-tool/>
 - <http://www.williammalone.com/articles/create-html5-canvas-javascript-drawing-app/>



Efecte

- Color Filter
 - Red filter: $r' = r; g' = 0; b' = 0$
- Negative / Invert
 - $r' = 255 - r; g' = 255 - g; b' = 255 - b;$
- Tonuri de gri / Greyscale
 - $r' = g' = b' = (r + g + b) / 3$
 - $r' = g' = b' = 0.299 * r + 0.587 * g + 0.114 * b$

Efecte

- Luminozitate / Brightness
 - $r' = r + \text{value};$ if ($r' > 255$) $r' = 255$ else if ($r' < 0$) $r' = 0;$
 - $g' = g + \text{value};$ if ($g' > 255$) $g' = 255$ else if ($g' < 0$) $g' = 0;$
 - $b' = b + \text{value};$ if ($b' > 255$) $b' = 255$ else if ($b' < 0$) $b' = 0;$
- Threshold
 - $v = (0.2126*r + 0.7152*g + 0.0722*b) \geq \text{threshold} ? 255 : 0; r' = g' = b' = v$

Efecte

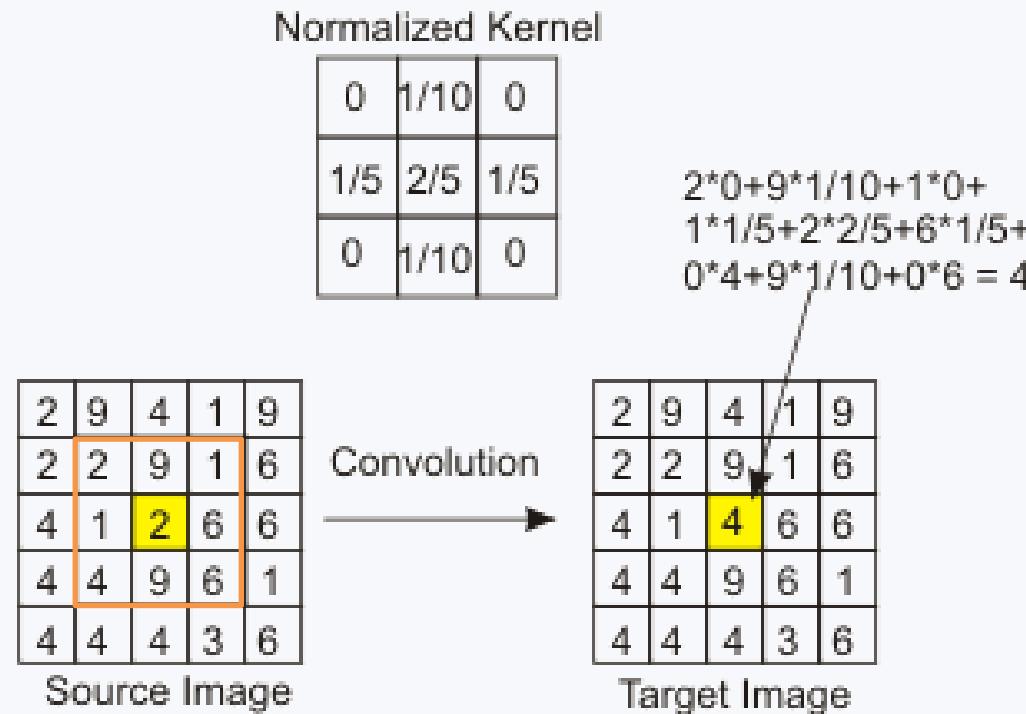
- Example:
 - <https://www.html5rocks.com/en/tutorials/canvas/imagefilters/#toc-simplefilters>
- Bibliografie suplimentară / exemple:
 - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Pixel manipulation with canvas

Efecte – Filtre de conoluție / Convolution filters

- Permit implementarea de filtre generice pentru procesarea imaginilor : blurring, sharpening, embossing, edge detection.
- Calculează noua valoare pentru un pixel pe baza valorilor pixelilor din apropiere din imaginea originală.
- **Exemplu:** blur filter
$$\begin{bmatrix} 1/9, 1/9, 1/9, \\ 1/9, 1/9, 1/9, \\ 1/9, 1/9, 1/9 \end{bmatrix}$$
- **Notă:** pentru a menține luminozitatea imaginii, suma valorilor din matrice ar trebui să fie egală cu unu.

Efecte – Filtre de conoluție / Convolution filters

- Example: <https://www.html5rocks.com/en/tutorials/canvas/imagefilters/#toc-convolution>



WebGL

- Documentation: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
- Three.js
 - cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser
 - Documentation: <https://threejs.org/>
 - Example: <https://heraclosgame.com/>

WebGL

- Babylonjs
 - JavaScript framework for building 3D games and experiences with HTML5, WebGL, WebVR and Web Audio
 - Documentation: <http://babylonjs.com/>

Grafică vectorială

Grafică vectorială

- În cazul **graficii raster**, imaginea este formată din pixeli dispuți pe rânduri și coloane, sub forma unei matrici.
- În cazul **graficii vectoriale** computerului i se oferă un **set de comenzi** pe care le execută pentru a desena imaginea. Imaginele conțin căi (en: paths) compuse din puncte, linii, curbe și forme (en: shapes).

```
<svg viewBox="0 0 220 100" xmlns="http://www.w3.org/2000/svg">
    <!-- Simple rectangle -->
    <rect width="100" height="100" />
    <!-- Rounded corner rectangle -->
    <rect x="120" width="100" height="100" rx="15" />
</svg>
```

Grafică vectorială

- Fiecare formă descrisă vectorială definește conturul unei regiuni geometrice care conține informații despre culoare.

```
<svg viewBox="0 0 220 100"  
      xmlns="http://www.w3.org/2000/svg">  
    <!-- Simple rectangle -->  
    <rect width="100" height="100" />  
    <!-- Rounded corner rectangle -->  
    <rect x="120" width="100" height="100" rx="15" />  
</svg>
```



Grafică vectorială



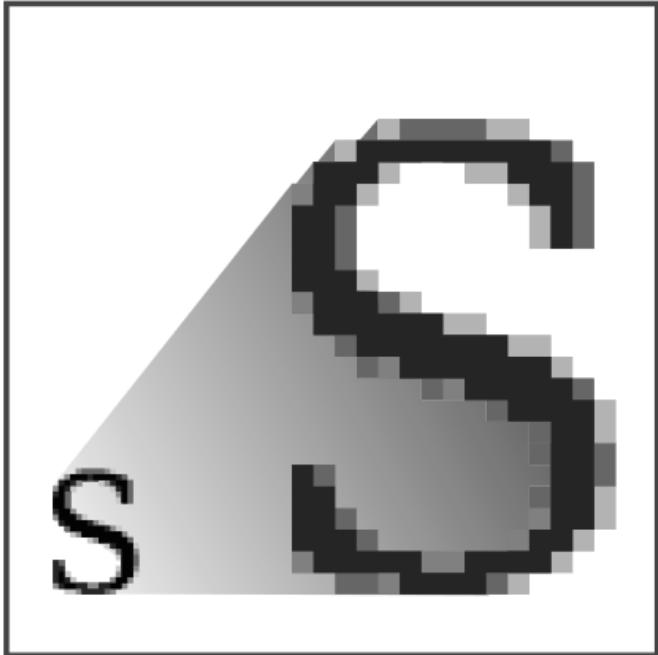
Grafică vectorială

- Deoarece căile (en: paths) pot fi redimensionate matematic, imaginile vectoriale pot fi redimensionate, fără a se pierde din claritatea imaginii.

```
<svg viewBox="0 0 220 100"  
xmlns="http://www.w3.org/2000/svg">  
    <!-- Simple rectangle -->  
    <rect width="100" height="100" />  
    <!-- Rounded corner rectangle -->  
    <rect x="120" width="100" height="100" rx="15" />  
</svg>
```

Redimensionare / Zoom

GIMP



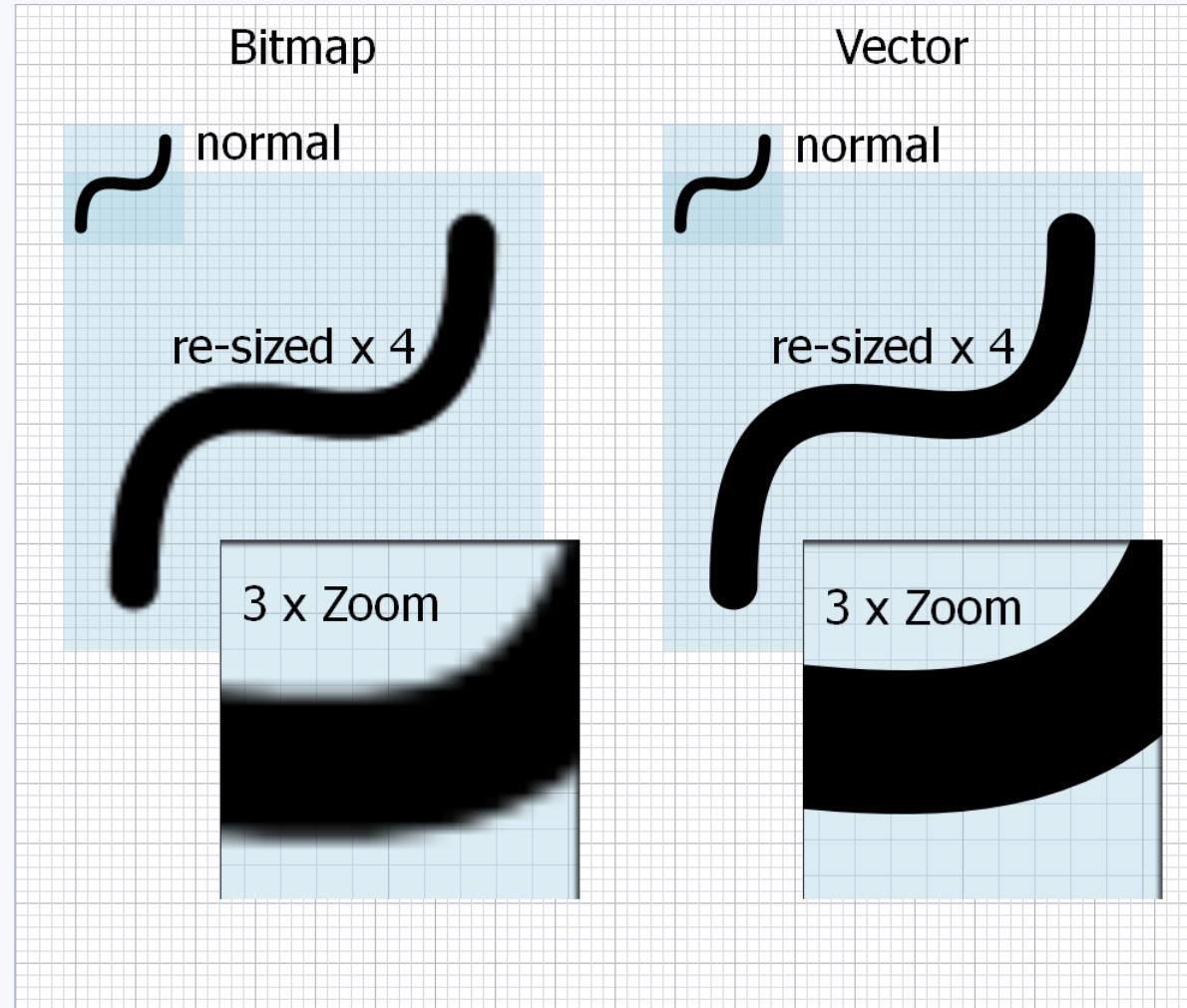
BITMAP
.jpeg .gif .png

INKSCAPE



OUTLINE
.svg

Redimensionare / Zoom



Animație

- Grafică vectorială poate fi utilizată pentru crearea de animații. În loc să desenăm fiecare cadru al proiectului separat (24 de cadre care apar în fiecare secundă) se pot utiliza instrumente software de animație care interpolatează matematic pozițiile componentelor în cadrele intermediare.



Avantaje și dezavantaje

- Avantaje :
 - Pentru imagini relativ simple, lista comenziilor utilizate pentru desenare ocupă mult mai puțin spațiu în fișier decât o versiune raster a aceleiași imagini.
 - Un program de desenare poate utiliza o comandă similară cu "RECT 300, 300, RED" pentru a desena un pătrat roșu cu laturile de 300 pixeli. Fișierul pentru această imagine conține 15 octeți care codifică informațiile alfanumerice din comandă.
 - Aceeași imagine ar putea fi creată și cu un program de grafică raster. Utilizând o adâncime de culoare de 8-bit (un byte per pixel), fișierul ar avea o dimensiune necomprimată de 90,000 bytes (300×300). Dimensiunile mult mai mici ale fișierelor vectoriale pot oferi beneficii semnificative în cazul aplicațiilor multimedia.

Avantaje și dezavantaje

- Imaginele vectoriale pot fi **redimensionate** fără pierderea calității. Imaginele vectoriale sunt mărite modificând parametrii formelor componente. Noua imagine poate fi apoi redesenată cu precizie la dimensiuni mai mari, fără distorsiunile tipice ale graficelor bitmap-uri mărite.
- Imaginele care sunt necesare în mai multe dimensiuni, cum ar fi sigla companiei, sunt adesea reprezentate cel mai bine cu ajutorul graficii vectoriale.

Avantaje și dezavantaje

- Dezavantaje:
 - Principalul dezavantaj al graficii desenate vectorial este lipsa controlului asupra pixelilor individuali ai imaginii. Prin urmare grafica vectorială nu este potrivită pentru afișarea și editarea imaginilor foto-realiste.

Formate

- SVG (Scalable Vector Graphics)
 - utilizează formatul XML pentru imagini bidimensionale
 - oferă suport pentru interactivitate și animație
- DXF (Drawing Exchange Format)
 - este un format CAD dezvoltat de Autodesk pentru a permite interoperabilitatea între AutoCAD și alte programe.

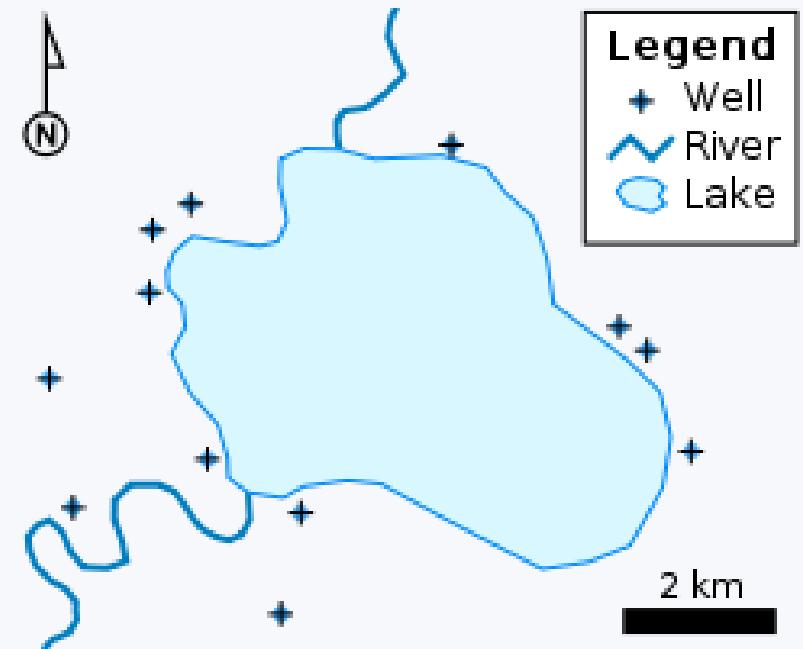
Formate

- EPS (Encapsulated Post Script)
 - creat de Adobe Systems pentru reprezentarea graficii vectoriale
 - utilizează un limbaj numit Post Script
- SHP (Shapefile)
 - dezvoltat de Esri pentru reprezentarea informațiilor geografice în contextul aplicațiilor GIS

Formate

Harta vectorială simplă:

- puncte (foraje),
- polilinii (râuri),
- poligoane (lac).



SVG – Scalable Vector Graphics

- documentație: <https://developer.mozilla.org/en-US/docs/Web/SVG>
- format de imagine vectorială bazat pe XML pentru grafică bidimensională

```
<!DOCTYPE html>
<html>
<body>
    <svg width="400" height="180">
        <rect x="50" y="20" width="150" height="150"
style="fill:red;stroke:black;stroke-width:5;opacity:0.5">
    </svg>
</body>
```

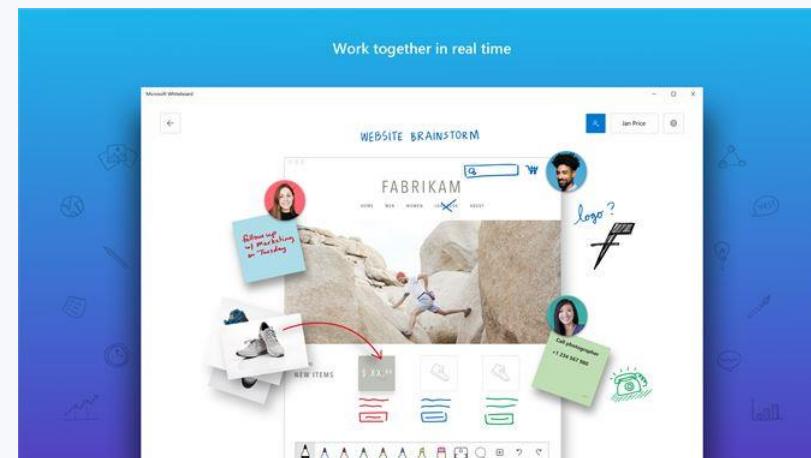
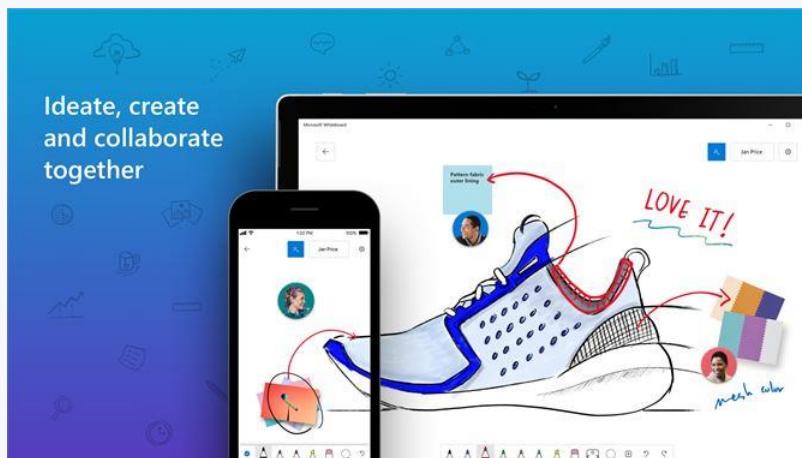
- oferă suport pentru interactivitate și animație

SVG – Scalable Vector Graphics



SVG – Scalable Vector Graphics

- Crearea imaginilor SVG
 - imaginile SVG pot fi create prin intermediul oricărui editor text, dar sunt mai ușor de construit cu ajutorul programelor de grafică vectorială precum [Inkscape](#).
 - Microsoft Whiteboard permite salvarea fișierelor rezultate în format SVG: [link](#).



SVG – Scalable Vector Graphics

- Linie

```
<line x1="start-x" y1="start-y" x2="end-x" y2="end-y">
```

- Example:

- http://www.w3schools.com/graphics/svg_line.asp

```
<svg height="210" width="500">
    <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />
</svg>
```

SVG – Scalable Vector Graphics

- Dreptunghi

```
<rect x="start-x" y="start-y" width="width" height="height" rx="radius-x"  
ry="radius-y"/>
```

- Example

- http://www.w3schools.com/graphics/svg_rect.asp

```
<svg width="400" height="110">  
  <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />  
</svg>
```

SVG – Scalable Vector Graphics

- Cerc

```
<circle cx="center-x" cy="center-y" r="radius"/>
```

- Example

- http://www.w3schools.com/graphics/svg_circle.asp

```
<svg height="100" width="100">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>
```

SVG – Scalable Vector Graphics

- Elipsă

```
<ellipse cx="center-x" cy="center-y" rx="radius-x" ry="radius-y"/>
```

- Poligon

```
<polygon points="x1,y1 x2,y2 ..."/>
```

- Polilinie

```
<polyline points="x1,y1 x2,y2 ..."/>
```

- Text

```
<text x="start-x" y="start-y">conținut</text>
```

SVG – Scalable Vector Graphics

- Grupare elemente

```
<g id="id_grup">... <!-- elemente --> </g>
```

- Definire grupuri

```
<defs>... <!-- definire grupuri --> </defs>
```

- Reutilizare grupuri

```
<use xlink:href="#id_grup" x="30" y="14"/>
```

Interacțiunea cu SVG utilizând CSS

- Documentație: [https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting started/SVG and CSS](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/SVG_and_CSS)
- Atribute de bază
 - stroke – culoare linie
 - stroke-opacity – opacitate linie
 - stroke-width – dimensiune linie
- fill – culoare suprafață
- fill-opacity – opacitate suprafață

Interacțiune cu SVG utilizând JavaScript / jQuery

- similar cu abordarea utilizată pentru elementele HTML;
- particularități:
 - atunci când creăm un element, trebuie să folosim spațiul de nume (en: namespace) SVG.

```
document.createElementNS("http://www.w3.org/2000/svg", „TAG_SVG”)
```

SVG – Scalable Vector Graphics

- Exemplu jQuery

```
$(document.createElementNS("http://www.w3.org/2000/svg", "rect"))  
    .attr({x:160, y:160, width:12, height:12})  
    .appendTo($("#drawing"));
```

Audio

Cuprins

1. Notiuni generale
2. Numerizarea sunetului
3. Formate audio
4. Compresia sunetului
5. Sunetul în context Web

Notiuni generale

- **Sunetul** este o vibrație propagată printr-un mediu material sub forma unei unde mecanice.
- Mediul este de obicei aer, deși sunetul poate fi transmis și prin solide și lichide.
- Exemplu: bătutul din palme produce un sunet prin comprimarea și deplasarea bruscă a moleculelor de aer. Perturbarea este transmisă moleculelor adiacente și se propagă prin spațiu sub forma unei unde. Auzim aplauzele mâinilor atunci când aceste vibrații ajung la urechile noastre.

Audio

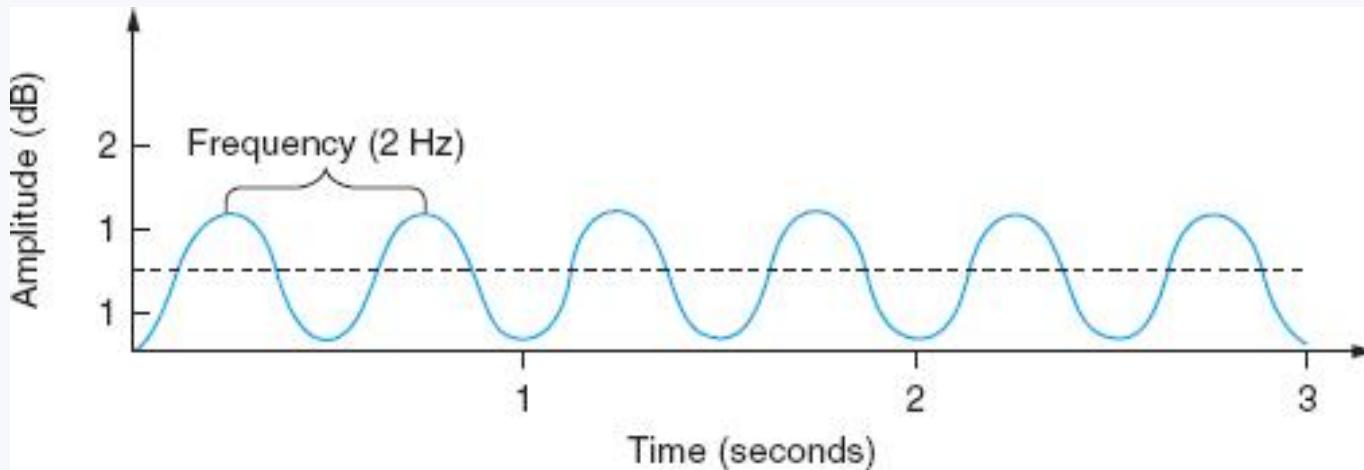
- Undele sonore sunt adesea comparate cu undele produse atunci când o piatră este aruncată într-un iaz. Piatra deplasează apa generând puncte înalte sau vârfuri de undă și puncte joase sau jgheaburi. Vedem aceste vârfuri și jgheaburi alternante ca modele de valuri care se deplasează spre exterior din punctul de impact al pietrei. Bătutul din plame produce, în mod similar, variații de presiune în aer. Aceste modificări ale presiunii aerului produc modele de unde care se răspândesc în toate direcțiile de la sursa sunetului.



Audio

- Urechea umană poate percepe sunete cu frecvență cuprinsă între aproximativ 20 Hz și 20 kHz
- **Zgomot:** caz particular de sunet caracterizat prin lipsa încărcăturii informaționale

Reprezentare

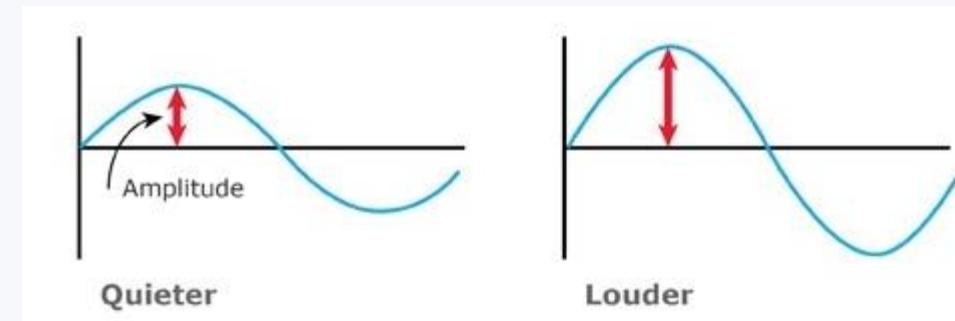


Caracteristici

- amplitudine
- frecvență
- durată

Caracteristici

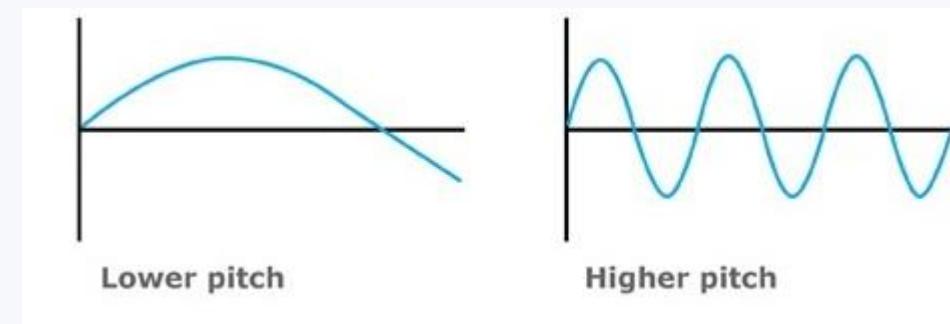
- **Amplitudinea** este o măsură a presiunii sonore sau a cantității de energie asociată sunetului. Aceasta este reprezentată de verticală sau axa y a undei sinusoidale. Amplitudinea este percepță ca **volum** al sunetului, care este de obicei măsurat în **decibeli** (dB).



- În general, sunetele cu amplitudini mai mari sunt mai puternice. Spectrul auzului uman este de la aproximativ 3 până la 140 dB. Fiecare creștere de 10 dB dublează aproximativ volumul percepțut al unui sunet.

Caracteristici

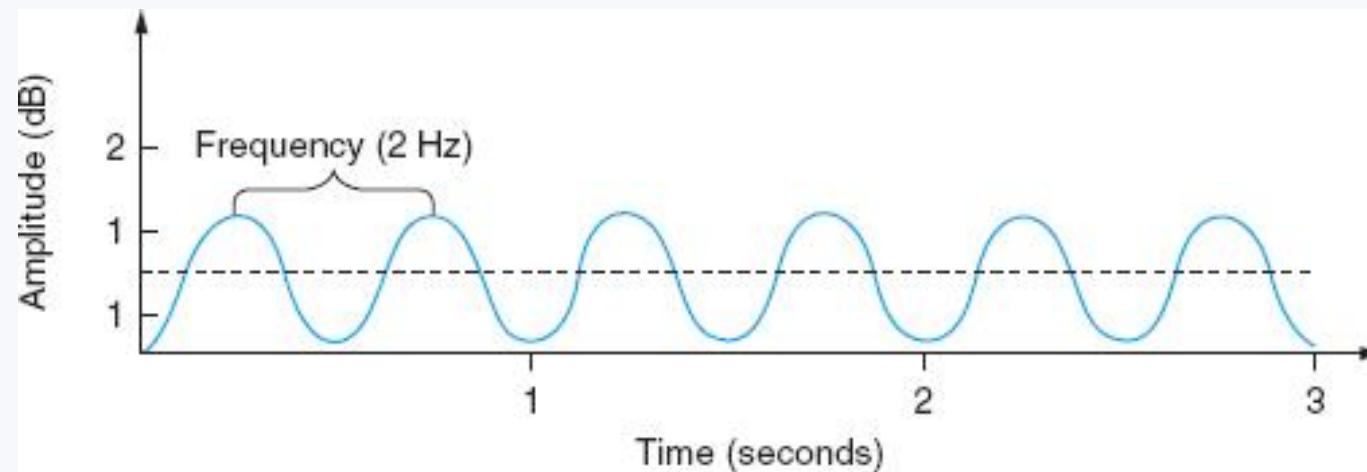
- **Frecvența** este numărul de repetări ale unei forme de undă într-un anumit interval de timp. Este reprezentată pe axa orizontală ca distanță între două vârfuri de undă sau jgheaburi. Frecvența este măsurată în **hertz** (Hz).
- Un **hertz** este **o repetare** a unei forme de undă într-o secundă de timp.



- Frecvențele înalte produc sunete de înăltime mai mare, iar frecvențele joase produc înăltimea joasă. Oamenii pot percepe un interval de frecvență de la 20 Hz la 20.000 Hz (sau 20 kilohertz (kHz), mii de hertz), deși majoritatea adulților nu pot auzi frecvențe peste 16 kHz.

Caracteristici

- Durata sunetului este echivalentă cu durata în timp a sunetului (ex: măsurată în secunde).

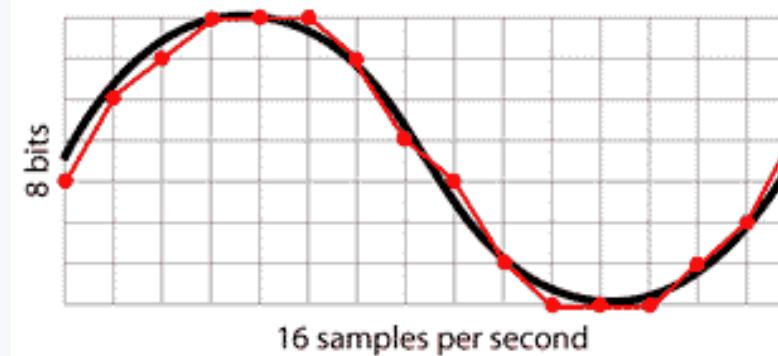
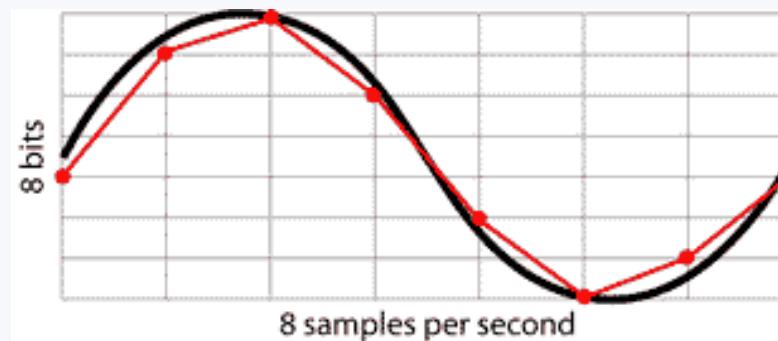


Audio Digital

- Tehnicile digitale reprezintă sunetul într-o formă discretă (discontinuă) de informații
- Există două tipuri majore de sunet digital:
 - *eșantionat*
 - *sintetizat*.

Audio Digital

- Sunetul eșantionat este o înregistrare digitală a undelor sonore analogice existente anterior. Un fișier pentru sunetul eșantionat conține multe mii de valori numerice, fiecare dintre acestea fiind o înregistrare a amplitudinii undei sonore la un moment particular, o eșantionare a sunetului.



Audio Digital

- **Sunetul sintetizat** este un sunet generat (sau sintetizat) de computer. Un fișier pentru sunetul sintetizat conține instrucțiuni pe care computerul le folosește pentru a produce sunetul.
- **Sunetul eșantionat** este de obicei utilizat pentru a capta și edita sunete naturale, cum ar fi vorbirea umană, spectacole muzicale și dramatice, ciripit de păsări, lansări de rachete și aşa mai departe. **Sunetul sintetizat** este utilizat în general pentru a crea compozitii muzicale originale sau pentru a produce efecte sonore noi.

Numerizarea sunetului

- presupune stocarea și prelucrarea sunetului în format digital
- sunetul este captat prin înregistrarea a numeroase măsurători separate ale amplitudinii unei unde utilizând un convertor ADC sau analog-digital. Un dispozitiv analogic, cum ar fi un microfon sau amplificatorul dintr-un sistem de difuzoare, generează un model de tensiune care variază continuu pentru a se potrivi cu unda sonoră originală. ADC eșantionează aceste tensiuni de mii de ori pe secundă. Aceste valori digitale sunt apoi utilizate pentru a recrea sunetul original prin conversia informațiilor digitale înapoi într-o formă analogică utilizând un convertor DAC sau digital-analog. DAC utilizează valorile amplitudinii pentru a genera variații de tensiune în curentul care alimentează difuzoarele pentru a reproduce sunetul

Numerizarea sunetului

Etape:

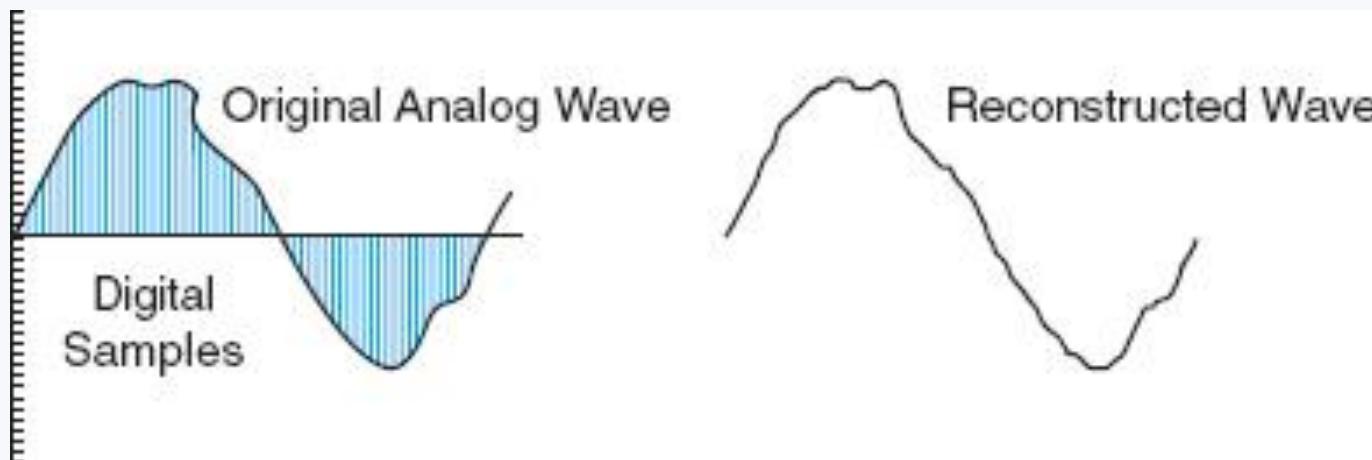
- Convertirea sunetului în semnal electric
- Eşantionarea şi cuantificarea semnalului
- Stocarea informaţiei numerice pe un suport de memorie externă conform unui format

Avantaje

- Stocare mai uşoară
- Permite analiza şi procesarea numerică a sunetului
- Nu se degradează în timp sau la copieri repetate

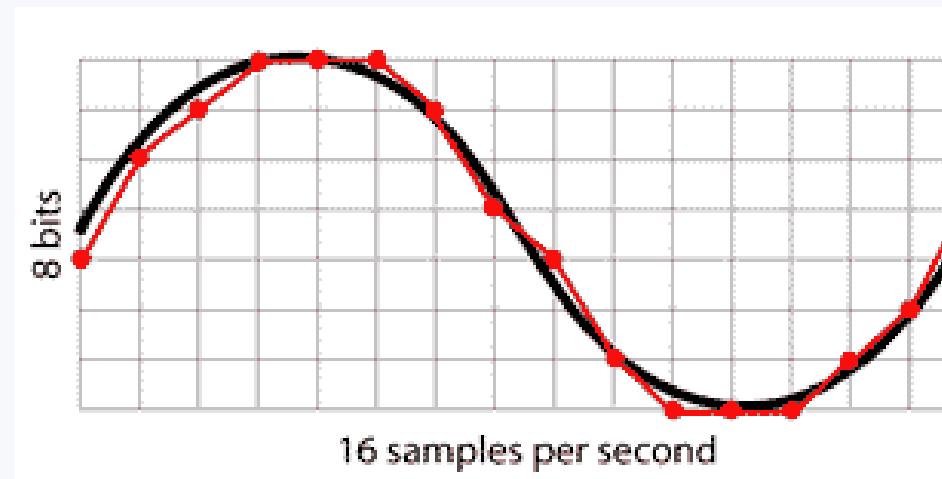
Numerizarea sunetului

- Eșantionarea digitală înlocuiește forma de undă continuă a sunetului original cu o nouă undă creată dintr-un număr fix de măsurători discrete.
- Unele informații se pierd în cadrul procesului de eșantionare, deoarece o undă continuă este infinit divizibilă și eșantionarea produce întotdeauna un număr finit de valori.
- Calitatea sunetului eșantionat depinde de doi factori conectați direct la acest proces de eșantionare: **rezoluția de eșantionare** (en: sample resolution) și **rata de eșantionare**.

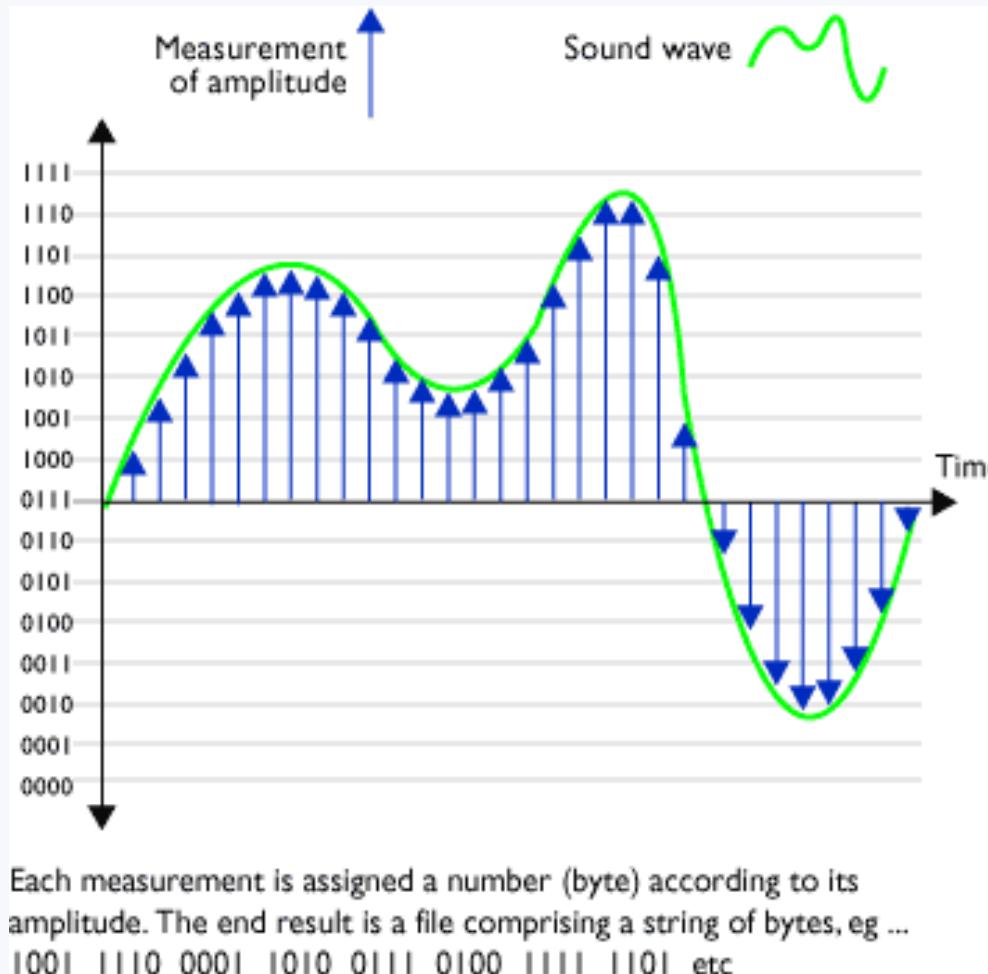


Numerizarea sunetului

- **Rezoluția de eşantionare** (en: sample resolution). Fiecare măsurare a amplitudinii efectuată de un ADC este înregistrată folosind un număr fix de biți. Numărul de biți utilizați pentru a codifica amplitudinea este cunoscut sub numele de rezoluție de eşantionare.
- Rezoluțiile de eşantionare variază între 8 și 32 de biți, cele mai frecvente fiind standardele CD-Audio pe 16 biți și standardele DVD-Audio pe 24 biți



Numerizarea sunetului



Numerizarea sunetului

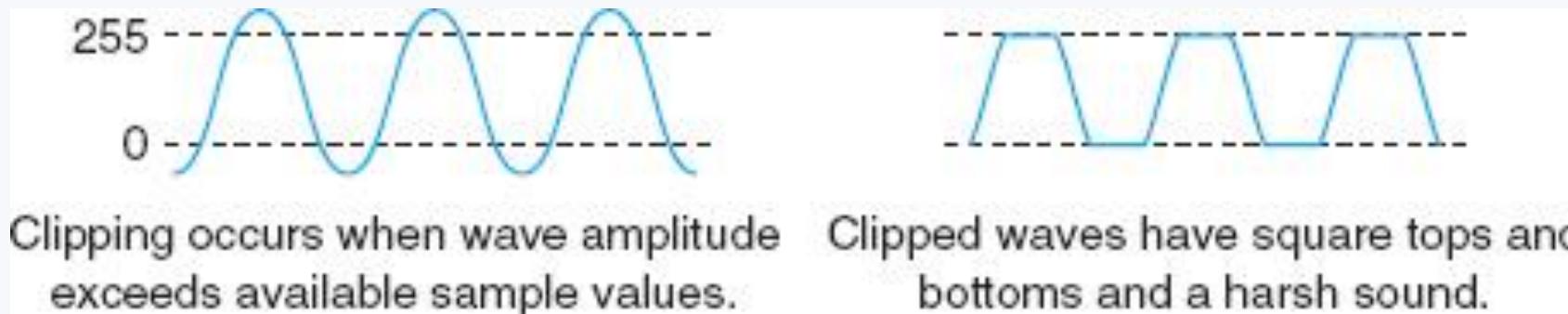
- Opt biți pot înregistra 256 de niveluri de amplitudine diferite. Acest lucru este adecvat pentru a surprinde un număr limitat de variații în amplitudine, cum ar fi cele dintre o șoaptă umană și un strigăt, dar sunt necesare rezoluții mai mari de eșantionare pentru a reproduce cu acuratețe sunete cu o gamă mai largă de amplitudini, cum ar fi spectacolele muzicale. Astfel, sunetul pe 8 biți este utilizat în general numai pentru sunete simple sau în aplicații multimedia care necesită dimensiuni foarte mici ale fișierelor.
- Standardul CD-Audio, care utilizează o rezoluție de eșantionare de 16 biți, acceptă peste 65 de mii de niveluri de amplitudine diferite, în timp ce standardul DVD-Audio pe 24 de biți poate reprezenta peste 16 milioane de niveluri.

Numerizarea sunetului

- Inadequate sample resolution can distort sound in two different ways: **quantization** and **clipping**.
- **Quantization** - each amplitude sample must be assigned one of the numbers available in the code being used. If the number of distinct values is too small, perceptually different amplitudes will be assigned the same number. Rounding a sample to the closest available value is known as **quantization**. In the case of sound, excessive quantization may produce a background hissing or a grainy sound. The solution is to record with a higher sample resolution (for instance, by using 16 rather than 8 bits).

Numerizarea sunetului

- **Clipping** - sound-sampling equipment is designed for a selected decibel range. If the source sound exceeds this range (as, for instance, when someone yells into a microphone held close to their lips), higher amplitudes cannot be encoded, because no values are available to represent them. The waveform of a clipped sound shows square tops and bottoms marking the point at which the highest amplitudes could not be captured. Clipping can produce a harsh, distorted sound.



Numerizarea sunetului

- The solution to **clipping** is to lower the amplitude of the source sound to record within the limits of the ADC circuitry.
- Recording equipment usually includes some form of meter such as a swinging needle, colored bars, or lights to show input levels and alert users when the amplitude range has been exceeded.
- The familiar, “Testing—one, two, three,” is often used to establish the proper distance and speech level when recording with a microphone.

Numerizarea sunetului

- Clipping can also occur during the mixing of different audio tracks.
- **Mixing** is the process of combining two or more sound selections, or *tracks*, into a single track. For example, a background music track might be mixed with a voice track of a poetry reading. This combination of two or more tracks may produce an amplitude that exceeds the available range. Adjustments to the volume of each track can eliminate the problem.
- Another solution is to use higher sample resolutions (for instance, 24-bit) to provide a wider range of amplitude values.

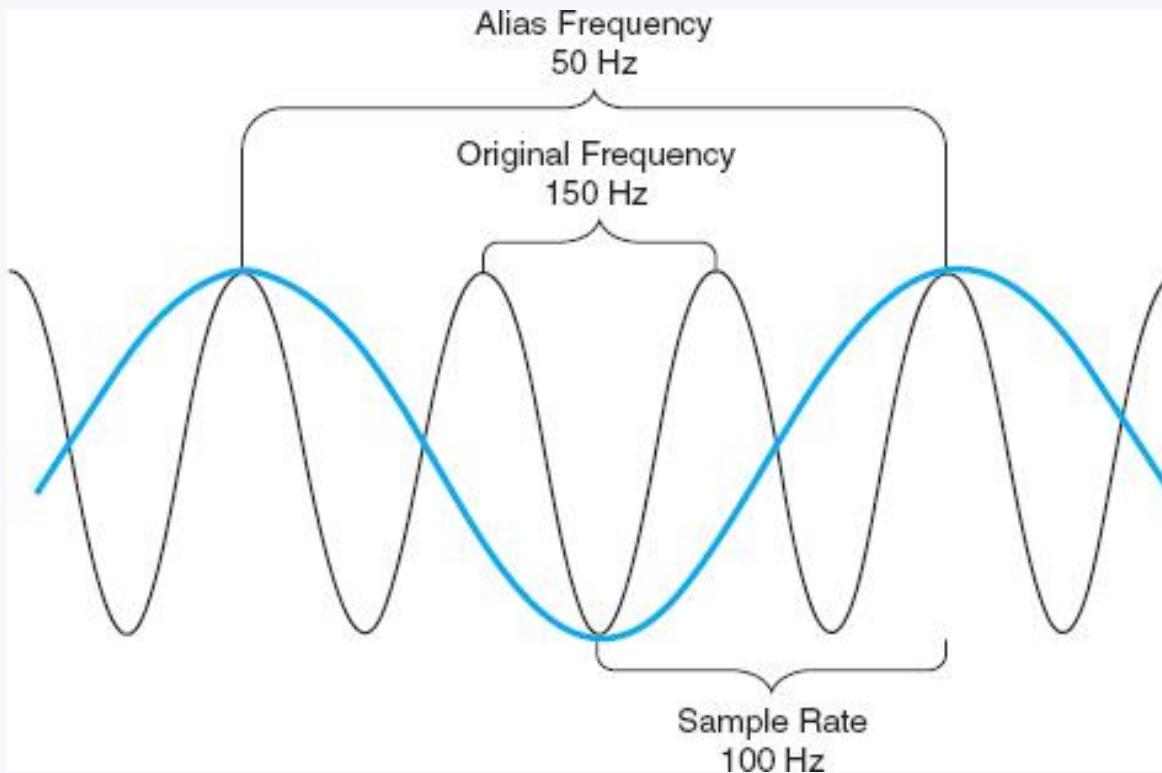
Numerizarea sunetului

- Frecvența de eşantionare (en: sample rate) este reprezentată de numărul de măsurători realizate într-un interval de timp fix. O rată de un eşantion pe secundă este desemnată ca Hertz.
- Deoarece eşantioanele de sunet sunt preluate întotdeauna de mii de ori pe secundă, ratele de eşantionare sunt de obicei indicate în kilohertz.
- Se determină pe baza teoremei lui Nyquist (minim dublul frecvenței maxime a sunetului)
- Frecvența de eşantionare afectează calitatea sunetului prin determinarea gamei de frecvențe care pot fi reprezentate într-o înregistrare digitală. Sunt necesare cel puțin două măsurători pentru a captura fiecare ciclu al unei unde sonore - una pentru fiecare valoare ridicată sau vârf și una pentru fiecare valoare scăzută sau jgheab. Cea mai mare frecvență care poate fi capturată este, astfel, jumătate din rata de eşantionare.

Numerizarea sunetului

- Sunetul de calitate CD captează 44.100 de eșantioane pe secundă (rata de eșantionare de 44,1 kHz) și poate reprezenta frecvențe de până la 22,050 Hz sau 22,05 kHz. DVD-Audio folosește o frecvență de eșantionare de 96 kHz pentru a capta frecvențe de până la 48 kHz.
- Sunetele care nu conțin frecvențe înalte pot fi reprezentate mai eficient folosind rate de eșantionare mai mici, deoarece acest lucru va produce o dimensiune totală mai mică a fișierului.
- O problemă potențială în cazul utilizării unei frecvențe de eșantionare mai reduse este aliasing-ul.

Numerizarea sunetului



Balancing File Size and Sound Quality

- It is not always necessary to use DVD-quality sound. For example normal speech, contain relatively low frequencies and a limited range of amplitudes. In this case, higher sample rates and sample resolutions do not improve sound quality, but they create needlessly large files. Using a rate of 11.025 kHz for voice recording results in a file only one-quarter the size of a CD-quality sound file. In addition, 8-bit sample resolution and monaural sound are usually adequate for speech. This further reduces file size by a factor of four, one-sixteenth the size of a stereo CD recording.

Resolution	Rate	Stereo/Mono	Size (1 Minute)	Quality
16	44.1 kHz	Stereo	10 MB	CD
16	22.05 kHz	Stereo	5 MB	FM radio
8	11.025 kHz	Mono	650 KB	AM radio
8	5.5 kHz	Mono	325 KB	Bad telephone

Compresia sunetului

- Reducerea **rezoluției de eşantionare** și a **frecvenței de eşantionare** sunt două moduri de a reduce dimensiunea unui fișier de sunet eşantionat. Aceste metode funcționează bine pentru sunete la frecvențe relativ joase și intervale de amplitudine înguste. Nu sunt eficiente pentru sunetele care conțin intervale mai largi de frecvență și amplitudine, cum ar fi spectacolele muzicale. În aceste cazuri poate fi utilizată o altă strategie: **compresia**.
- Compresia poate fi fie fără pierderi, fie cu pierderi. **Compresia fără pierderi** utilizează o codificare mai eficientă pentru a reduce dimensiunea unui fișier, păstrând în același timp toate informațiile din original. **Compresia cu pierderi** renunță la o parte din informațiile originale.

Compresia sunetului

- Deoarece algoritmii de **compresie cu pierderi** produc fișiere mult mai mici, acestea sunt tehnica preferată pentru compresia sunetului.
- Codecurile de compresie a sunetului cu pierderi folosesc diverse tehnici pentru a reduce dimensiunile fișierelor. Unele dintre acestea profită de psihacoacustică, de interacțiunea dintre condițiile psihologice ale percepției umane și proprietățile sunetului. De exemplu, în timp ce oamenii cu auz optim pot percepe frecvențe de până la aproximativ 20 kHz, majoritatea oamenilor nu pot distinge frecvențe peste aproximativ 16 kHz. Aceasta înseamnă că informațiile cu frecvență mai mare pot fi eliminate fără ca majoritatea ascultătorilor să observe o diferență. Sunetele cu amplitudine mai mare într-un canal stereo vor „acoperi”, de obicei, sunete mai slabe prezente pe celălalt canal.

Compresia sunetului

1. Mascarea frecvențelor

- Sunt eliminate sunetele cu frecvență mai mare de 16-18 KHz
- Sunt eliminate sunetele de intensitate scazută, care apar concomitent cu sunete de intensitate înaltă, dacă sunt în benzi de frecvență alăturate (cele cu intensitate scazută sunt măcate de cele cu intensitate înaltă)

2. Mascarea temporală

- Se elimină sunetele de intensitate mică care urmează după sunete de intensitate puternică în cadrul unui interval de timp. Sunetele de intensitate mică nu pot fi percepute după sunete de intensitate puternică datorită inerției timpanului.

Compresia sunetului

- Compresia cu pierderi folosește, de asemenea, alte tehnici, cum ar fi cuantificarea cu **rată de biți variabilă (VBR)**. În VBR, sunetele sunt codificate folosind un număr diferit de biți pe secundă, în funcție de complexitatea sunetului. Pentru pasaje simple de sunet cu frecvențe limitate, se folosește un număr mai mic de biți pe secundă decât pentru pasaje mai complexe, cum ar fi cele cu multe instrumente diferite și frecvențe mai mari.
- Algoritmii de codificare cu pierderi de informație, cum ar fi MP3, pot reduce dimensiunile fișierelor cu până la 80%, păstrând o calitate similară cu cea a unui CD audio.

Formate audio

- MP3 (MPEG1, audio layer 3)
 - extenise: .mp3 or .mpga
 - format audio popular care oferă o compresie semnificativă păstrând în același timp o calitate excelentă.
 - utilizat frecvent în aplicații web.

Formate audio

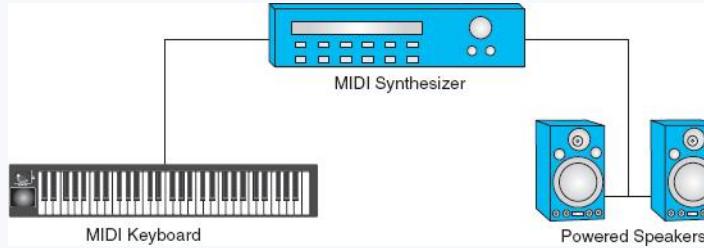
- AAC (advanced audio coding)
 - extensie: .mp4 and .aac
 - succesorul MP3 specificat în standardul MPEG4, care oferă o calitate a sunetului mai bună decât MP3 la rate de biți comparabile.
 - de asemenea, poate reduce semnificativ dimensiunile fișierelor pentru o calitate audio comparabilă.
 - utilizat de: Apple iPod, Apple iPad, Apple iPhone, YouTube, and Sony PlayStation.

Formate audio

- Alte formate de fișiere audio:
 - WMA (Windows Media Audio);
 - WAVE - formatul standard de fișier audio ce conține sunet în reprezentare PCM necomprimat;
 - AIFF (Audio Interchange File Format) - formatul standard pentru audio digital utilizat pe platformele Apple (variante: necomprimat / comprimat);
 - AU.

Audio

Synthesized Sound



Further reading: <https://en.wikipedia.org/wiki/MIDI>

Web Audio - <audio>

<audio> - elementul HTML este utilizat pentru a încorpora conținut sonor în aplicații web. Poate conține una sau mai multe surse audio, reprezentate folosind atributul "src" sau elementul <source>, dintre care browser-ul îl va alege pe cel mai potrivit.

```
<audio controls>
```

```
  <source src="media/viper.mp3" type="audio/mp3">
```

```
  <source src="media/viper.ogg" type="audio/ogg">
```

```
  <p>Your browser doesn't support HTML5 audio.</p>
```

```
</audio>
```



- Încercați codul de mai sus în Internet Explorer, comentând formatul mp3

Web Audio

- Atributele elementului `<audio>`:
 - `autoplay (bool)` – redarea automată a sunetului
 - `controls (string)` – controalele de redare sunt afișate dacă atributul este prezent
 - `loop (bool)` – permite redarea continuă a sunetului
 - `src (string)` – permite specificarea sursei fără utilizarea de tag-uri de tip `source`
- Elementul `<source>`
 - `src (string)` – adresa (URL) fișierului audio
 - `type (string)` – tipul MIME pentru fișierul audio
- API: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio>

Web Audio - HTMLAudioElement

HTMLAudioElement (JavaScript) - oferă acces la proprietățile elementelor <audio>, precum și la metodele aplicabile acestora. Este derivat din HTMLMediaElement.

- Proprietăți:
 - src / currentSrc – URL-ul absolut al fișierului redat
 - currentTime – poziția (în secunde) în cadrul fișierului (poate fi modificată)
 - duration – durata totală a fișierului audio (în secunde)
 - ended – boolean setat pe true la terminarea redării
 - error – ultima eroare (obiect MediaError) sau null dacă nu a apărut nici o eroare
 - paused – boolean setat pe false la oprirea redării

Web Audio - HTMLAudioElement

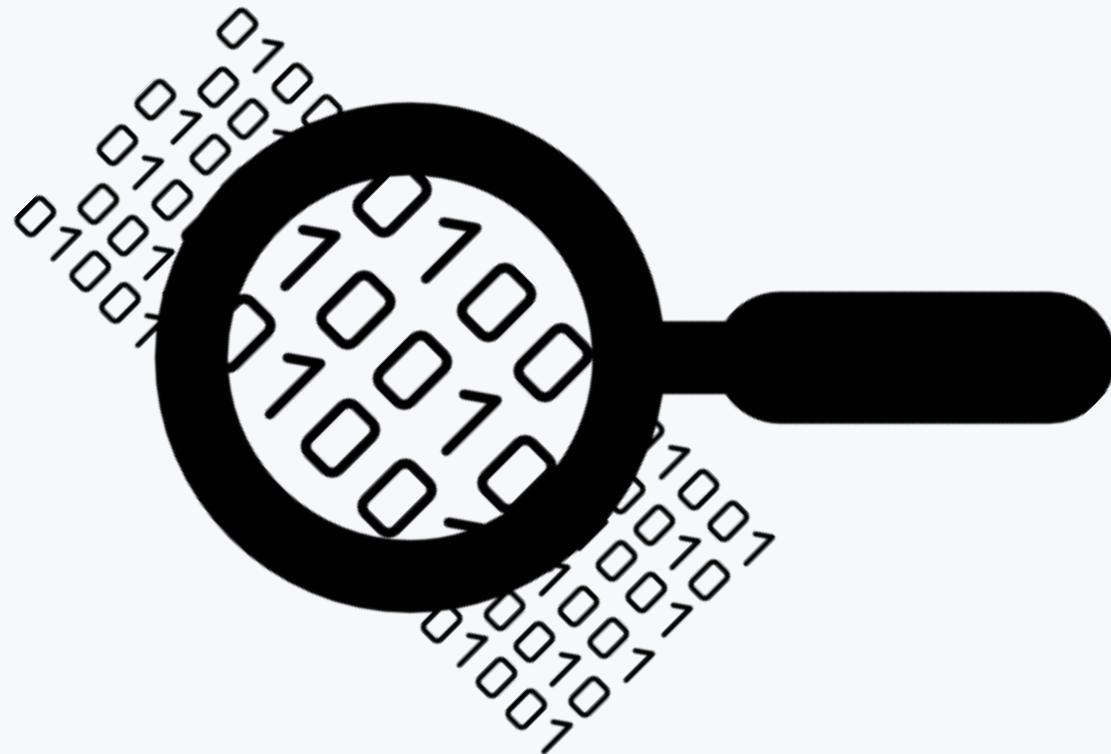
- Proprietăți (continuare):
 - readyState – indică starea curentă a elementului
 - volume – permite citirea / modificarea volumului
- Metode:
 - canPlayType(type) – permite aplicației să determine dacă browser-ul curent suportă un anumit tip de fișier audio
 - load() – pornește procesul de descărcare a fișierului audio de pe server; este obligatoriu să fie apelat înainte de începerea redării folosind metoda play()
 - pause() – oprește redarea (cu păstrarea poziției curente)
 - play() – pornește redarea de la poziția curentă

Web Audio - HTMLAudioElement

- Evenimente:
 - canplay – a fost încărcată o parte din fișier și poate fi pornită redarea
 - ended – redarea s-a terminat
 - pause – redarea a fost oprită
 - play – redarea a început
 - volumechange – modificare de volum
 - waiting – operația curentă este suspendată pentru a încărca date de pe server
- API: <https://developer.mozilla.org/en-US/docs/Web/API/HTMLAudioElement>

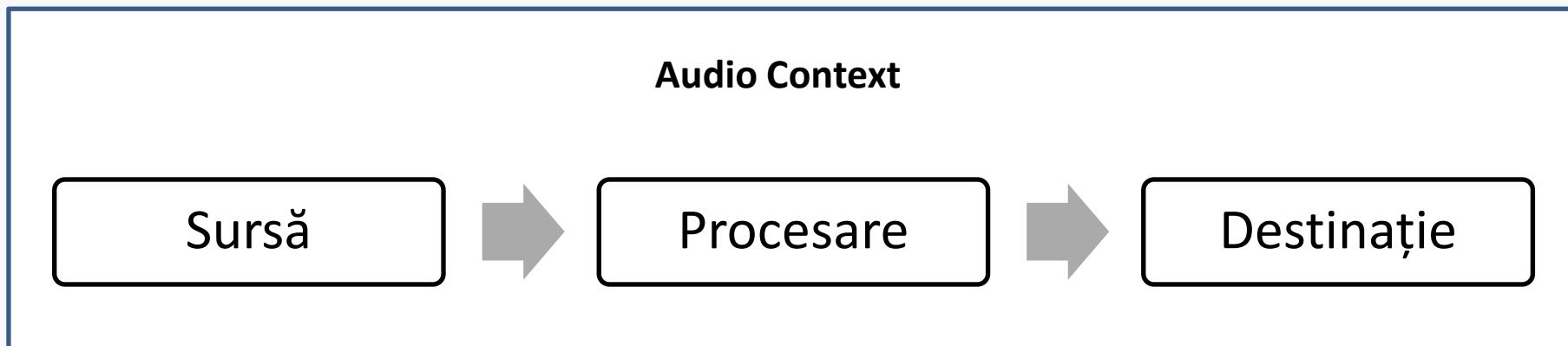
Lab Example:

- <https://ase-multimedia.azurewebsites.net/audio-playlist/>



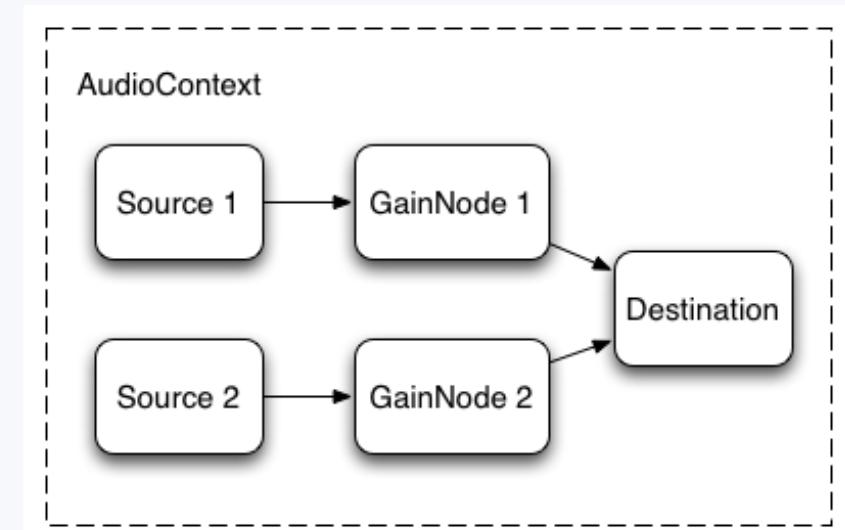
Web Audio API

- Web Audio API - oferă facilități avansate pentru lucrul cu fișiere audio pe web, permitând dezvoltatorilor să aleagă surse audio, să adauge efecte audio, să creeze vizualizări ale sunetului.
- API: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API



Web Audio API

- Operațiile:
 - se desfășoară în interiorul unui context audio
 - sunt specificate prin intermediul unui graf de rutare
- Tipuri de noduri:
 - Sursă: preiau datele audio dintr-un element <audio>, de la microfon sau generează sunete
 - Procesare: prelucrează datele primite la intrare
 - Destinație: nodul final al graficului



Web Audio API

AudioContext—reprezintă un graf de procesare audio

- Proprietăți: `.currentTime` (poziția curentă în secunde), `.destination` (referință la nodul destinație)
- Metode:
 - **createMediaElementSource(audioElement)**: construiește un nod sursă pe baza unui element `<audio>`
 - **createMediaStreamSource(stream)**: construiește un nod sursă pe baza unui stream
 - **createOscillator()**: construiește un nod sursă pentru generarea de sunete
 - **createGain()**: construiește un nod de procesare pentru ajustarea volumului
 - **createAnalyser()**: permite analiza sunetului (descompunere Fourier)
 - **createScriptProcessor(bufferSize)**: construiește un nod pentru procesare JavaScript
 - **close()**: închide audiocontext-ul, eliberând orice resurse audio de sistem utilizate.

AudioNode—reprezintă un nod din cadrul grafului (din care sunt moștenite celelalte noduri)

- Metode:
 - `connect(node)`: conectează ieșirea nodului curent la intrarea nodului primit ca parametru
 - diverse metode și proprietăți în funcție de tipul nodului

Web Audio API



```
const context = new AudioContext();

navigator.mediaDevices.getUserMedia({ audio: true })
.then((stream) => {
  const microphone = context.createMediaStreamSource(stream);
  const filter = context.createBiquadFilter();
  // microphone -> filter -> destination
  microphone.connect(filter);
  filter.connect(context.destination);
});
```

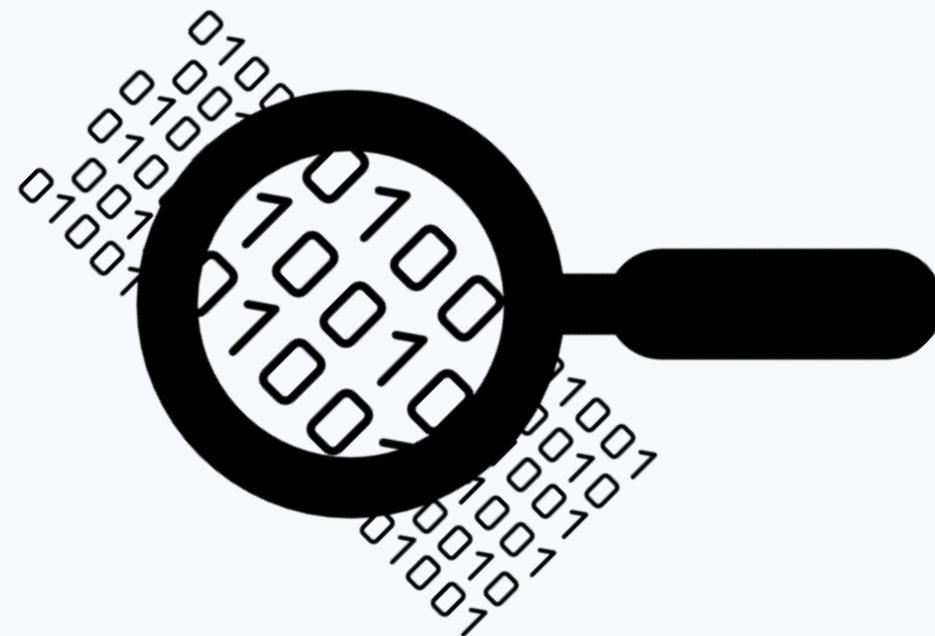
Test the code on this slide in your browser

Web Audio API

- Creare de vizualiări: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Visualizations_with_Web_Audio_API
- Record and save as MP3:
<https://www.html5rocks.com/en/tutorials/getusermedia/intro/>
- Playground: <http://webaudioplayground.appspot.com/>

Lab Example: Web Audio API

- <https://ase-multimedia.azurewebsites.net/audio-web-audio-api/>



WebRTC API

- WebRTC (Web Real-Time Communications) este o tehnologie care permite aplicațiilor și site-urilor web să capteze și să transmită conținut audio și / sau video.
- Setul de standarde care cuprinde WebRTC face posibilă efectuarea de teleconferințe peer-to-peer, fără a fi necesar ca utilizatorul să instaleze plugin-uri sau orice alt software terță parte.
- Web RTC API: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API,
<https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia>

Demo

- https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Taking_still_photos



Lab Example: Speech API

- <https://ase-multimedia.azurewebsites.net/speech-api/>



Alte exemple

- <https://googlechromelabs.github.io/web-audio-samples/>

Recommendări

- În general nu este recomandată utilizarea atributului "autoplay"
- Este necesar ca utilizatorii:
 - Să poată controla volumul sunetului.
 - Să poată opri / porni redarea sunetului.
- Se va evita utilizarea excesivă a sunetului. Sunetul poate fi mai obositor pentru utilizatori decât imaginile sau textul.

Video digital

Definition

- Cuprinde totalitatea tehnicilor de captură, procesare și stocare a imaginilor în mișcare (precum și a sunetului asociat) prin intermediul unui dispozitiv de calcul.
- Video digital cuprinde o serie de imagini digitale afișate în succesiune rapidă. În contrast, filmul analogic, folosește o serie de fotografii afișate în succesiune rapidă.
- Video digital a fost introdus pentru prima dată comercial în 1986 cu formatul Sony D1.

Avantaje

- Poate fi procesat prin intermediul calculatorului
- Păstrare în timp și rezistență la copieri repetitive
- Poate fi transmis la distanță

Caracteristici video

- Rezoluția
- Spațiul de culoare și numărul de biți per pixel
- Numărul de cadre pe secundă
- Modul de afișare (interlaced sau progresiv)
- Metode de compresie

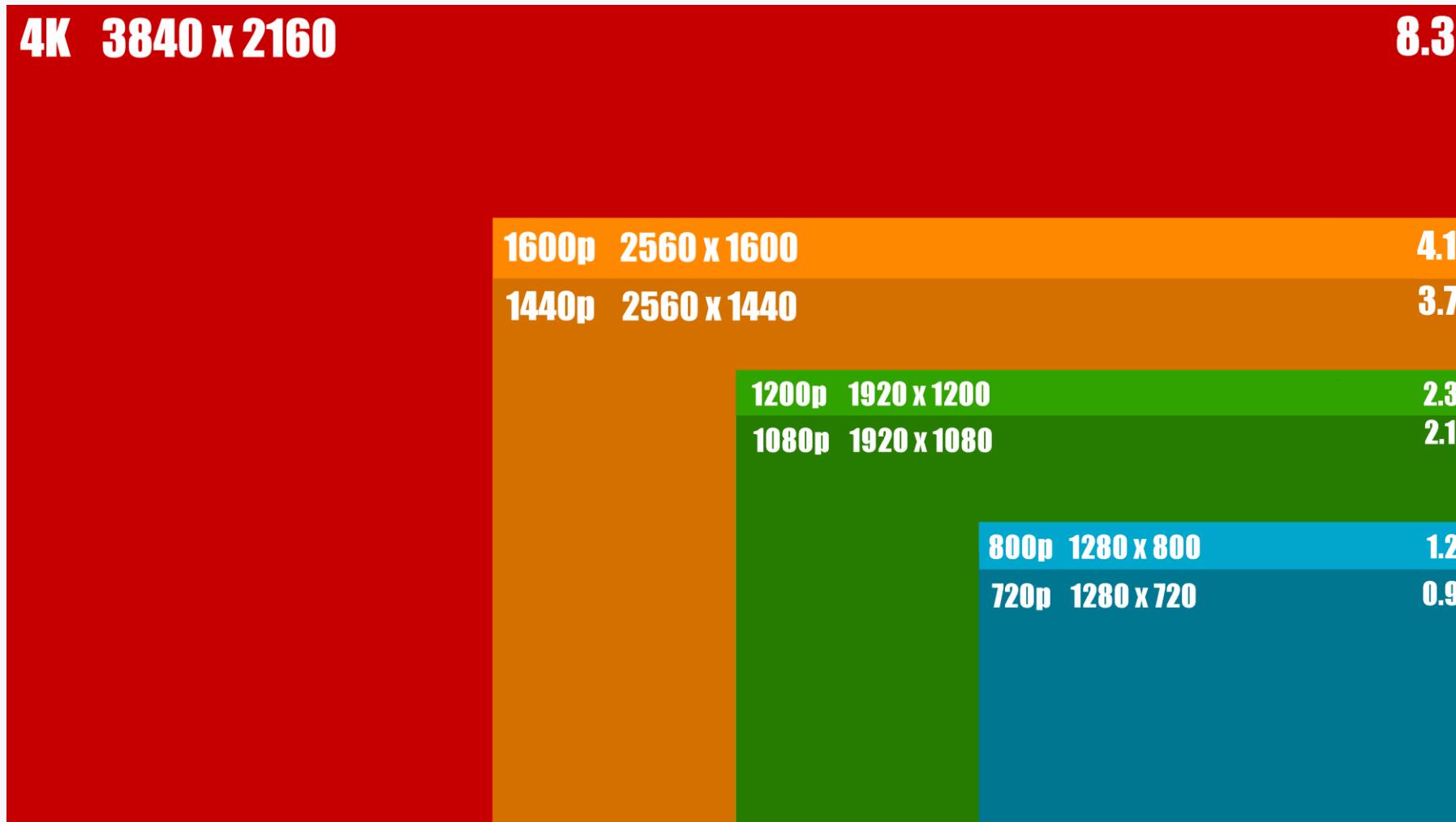
Caracteristici - Rezoluția

- **lățimea și înălțimea** imaginii afișate, măsurate în pixeli. Cu alte cuvinte, numărul total de pixeli conținuți în fiecare cadru individual (numit și **rezoluție spațială**)[1]
- cu cât rezoluția este mai redusă, cu atât puterea de procesare necesară și spațiul de stocare sunt mai reduse [2].
- Rezoluția formatului DV (video digital) este de **720 × 480 pixeli**. Este astfel necesar ca dispozitivul de redare să proceseze și să transmită informații pentru aproape **350,000 pixeli** la rate de până la 30 de ori pe secundă[2].

Characteristics – Common Screen Resolution

Name	Pixels (width x height)	Aspect Ratio	Notes
<i>Standard Definition (SD)</i>			
480p / 480i	720x480 (or 704x480)	4:3 (approx)	NTSC
576p / 576i	720x576 (or 704x576)	4:3 (approx)	PAL
<i>High Definition (HDTV)</i>			
720p	1280x720	16:9	
1080p / 1080i	1920x1080	16:9	
<i>Ultra High Definition (UHDTV)</i>			
4K (2160p)	3840x2160	16:9	Exactly 4 × 1080p
8K (4320p)	7680x4320	16:9	Exactly 16 × 1080p
8640p	15360x8640	16:9	Exactly 32 × 1080p
<i>Digital Cinema (DCI)</i>			
2K	2048 × 1080	1.90:1	The first generation of digital cinema projectors.
4K	4096 × 2160	1.90:1	2nd generation digital cinema.

Caracteristici – Rezoluție



Caracteristici – Rezoluție



Caracteristici - Spațiul de culoare și numărul de biți per pixel

- Spațiul de culoare este numărul de biți utilizați pentru a indica culoarea unui singur pixel sau numărul de biți utilizați pentru fiecare componentă de culoare a unui singur pixel
- Similar cu imaginile raster, cu cât sunt utilizați mai mulți biți pentru stocarea culorii, cu atât pot fi reproduse variații mai subtile ale culorilor.

Caracteristici – Numărul de cadre pe secundă

- Reprezintă frecvența la care un dispozitiv afișează imagini consecutive numite cadre.
- Percepția mișcării continue în videoclip este dependentă de doi factori. În primul rând, imaginile trebuie afișate suficient de rapid pentru a permite persistenței vederii să umple intervalul dintre cadre. În al doilea rând, schimbările în locația obiectelor de la cadru la cadru trebuie să fie graduale.

Caracteristici – Frame Rate

- O frecvență obișnuită pentru fișierele video este de 30 de cadre pe secundă (fps). Videoclipurile destinate streamingului pe Internet sunt adesea livrate la o rată de doar 15 fps, reducând efectiv volumul de date necesar la jumătate[1]
- NTSC utiliza aproximativ 30 de cadre pe secundă, în timp ce PAL utiliza 25 de cadre pe secundă [2].

Characteristics – Interlaced / Progressive

- **Progressive video**
 - each frame is displayed similar to the text on a page - line by line, top to bottom.
- **Interlaced video**
 - each frame is composed of two halves of an image. In the first pass all odd numbered lines are displayed, from the top left corner to the bottom right corner. The second pass displays the second and all even numbered lines, filling in the gaps in the first scan.
 - the two halves are referred to individually as fields. Two consecutive fields compose a full frame. If an interlaced video has a frame rate of 15 frames per second the field rate is 30 fields per second.

Characteristics – Interlaced / Progressive

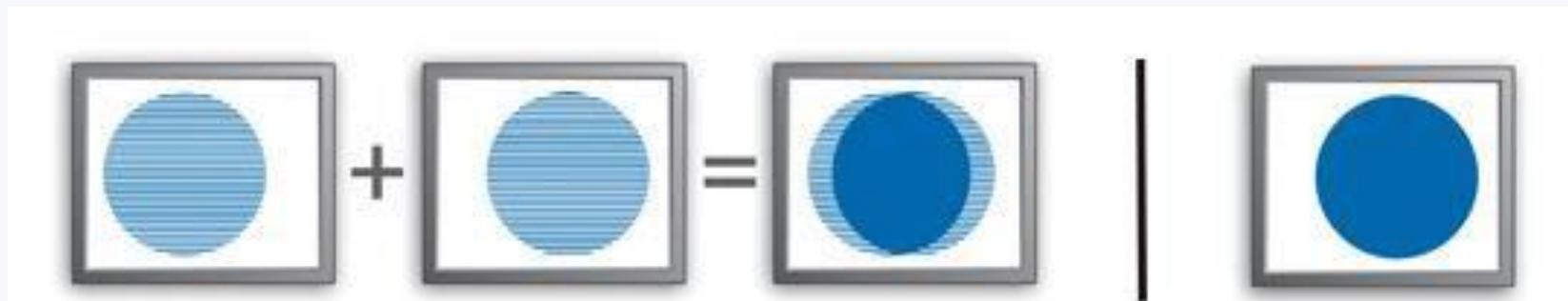


Characteristics – Interlaced / Progressive

- Examples of formats:
 - PAL -576p (progressive) / 576i (interlaced)
 - FullHD – 1080p (progressive) / 1080i (interlaced)
- Benefits of interlacing
 - for a fixed bandwidth, interlace provides a video signal with twice the display refresh rate for a given line count (versus progressive scan video at a similar frame rate—for instance 1080i at 60 half-frames per second, vs. 1080p at 30 full frames per second).
 - bandwidth benefits **only apply** to an **analog or uncompressed digital video** signal. With digital video compression, as used in all current digital TV standards, interlacing introduces additional inefficiencies.

Characteristics – Interlaced / Progressive

- Interlacing issues
 - Because each interlaced video frame is composed of two fields captured at different moments in time, interlaced video frames can exhibit motion artifacts known as interlacing effects, if recorded objects move fast enough to be in different positions when each individual field is captured.
 - These artifacts may be more visible when interlaced video is displayed at a slower speed than it was captured, or in still frames.



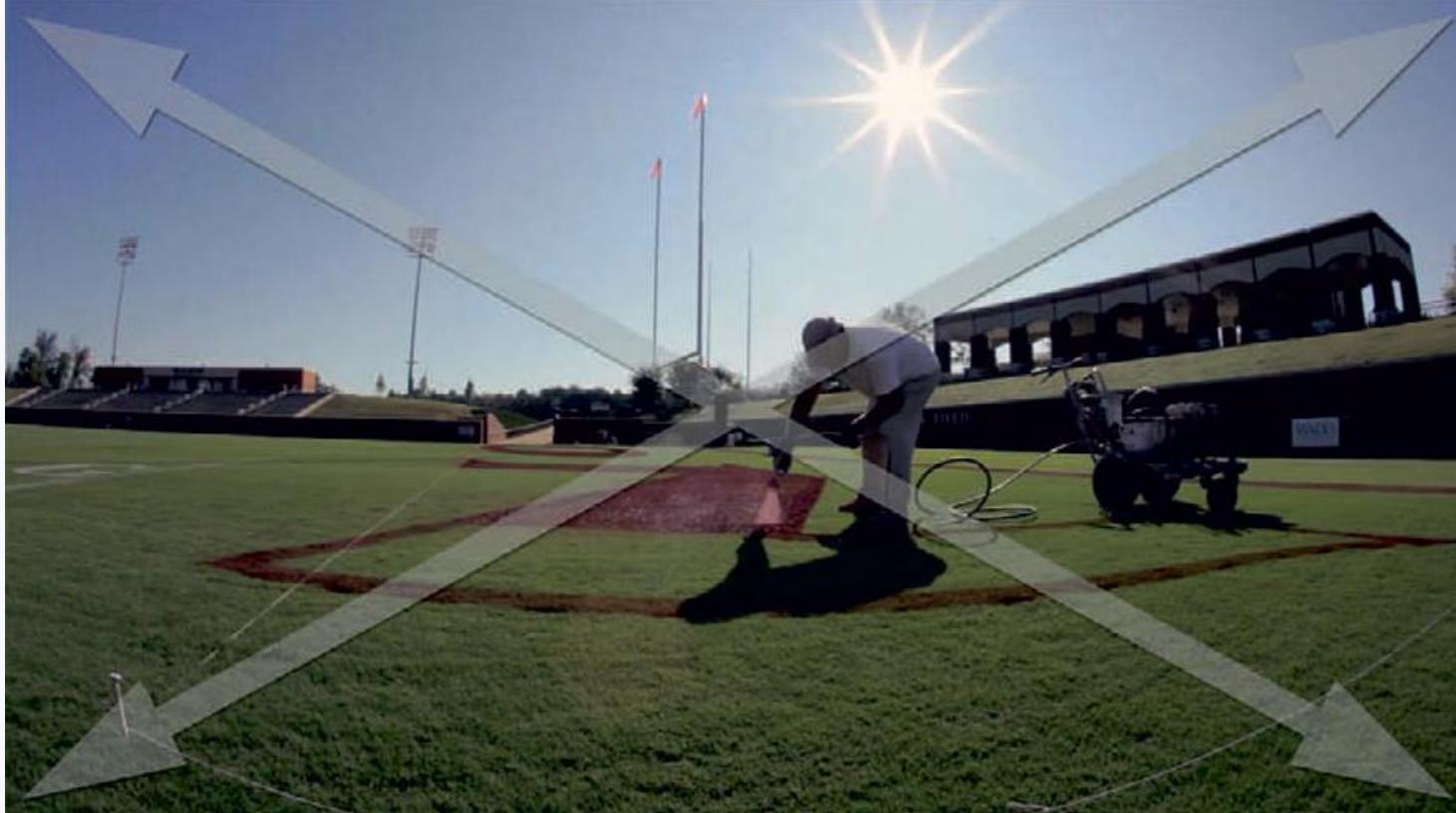
Caracteristici – Compresia

- Scopul compresiei video:
 1. reducerea dimensiunii fișierelor video prin eliminarea informației redundante.
 2. păstrarea calității percepute a imaginii.

Caracteristici – Compresia

- Redundanță poate apărea:
 - în fiecare cadru considerat individual (similar imaginilor) – *redundanță spațială*.
 - de-a lungul timpului într-o secvență video care conține multe cadre – *redundanță temporală*.
- **Exemplu:** o înregistrare de cinci secunde (150 de cadre) a unui cer albastru într-o zi fără nori.
 - mii de pixeli albaștri păstrează aceeași valoare a culorii pe întreaga secvență de 150 de cadre. Acest fenomen se numește **redundanță temporală** și apare ori de câte ori valoarea unui pixel rămâne neschimbată de la un cadru la altul într-o secvență bazată pe timp. De asemenea, pixelii mulți pixeli au aceeași culoare în interiorul fiecărui cadru. Acest fenomen se numește **redundanță spațială**

Caracteristici – Compresia



Redundanță spațială (intra-cadru)

Caracteristici – Compresia



Redundanță temporală (1 second / 30 Frames)

Caracteristici – Compresia

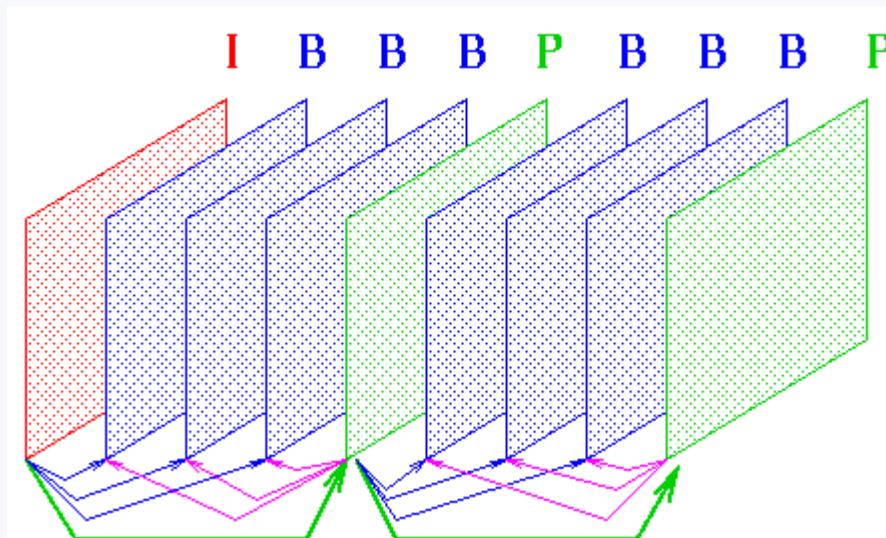
- *Compresie Intraframe (or I-frame)*
 - Elimină redundanța spațială „în interiorul” unui cadru video în același mod în care compresia JPEG este utilizată pentru a reduce redundanța într-o fotografie digitală.
 - I-frames sunt de obicei comprimate la un raport de 10:1. Aceasta înseamnă că un cadru I-frame comprimat consumă doar 10% din spațiul necesar unui cadru necomprimat.
 - Deoarece cadrele I-frame sunt complet definite de informațiile stocate, ele pot fi ușor decodate și redate pe ecran.

Caracteristici – Compresia

- **Compresie Interframe** (metoda cel mai frecvent utilizată)
 - exploatează atât redundanțele **spațiale**, cât și cele **temporale**
 - folosind metoda anterioară de compresie intraframe, toate cadrele dintr-un flux video sunt transformate în cadre de tip I-frame. Fiecare cadru este comprimat prin eliminarea redundanței spațiale. Astfel, compresia se aplică uniform tuturor cadrelor dintr-un flux video.
 - cu **compresie interframe**, cadrele I-frame sunt create la intervale fixe (de obicei la fiecare 15 cadre). Un cadru I-frame marchează începutul unei secvențe împachetate de cadre adiacente numite GOP (Group of Pictures). Cadrul I-frame complet definit servește drept cadru cheie sau referință pentru celelalte cadre din grup. Sarcina sa este să mențină valorile pixelilor care nu se vor schimba pe parcursul secvenței. Culoarea pixelilor se va modifica doar dacă această modificare este stocată explicit în cadrele următoare.

Caracteristici – Compresia

- Compresia MPEG
 - exploatează atât redundanțele spațiale, cât și cele temporale (compresie interframe)
 - fiecare cadru I-frame este urmat de o secvență de 14 cadre desemnate drept cadre B-frame sau P-frame.



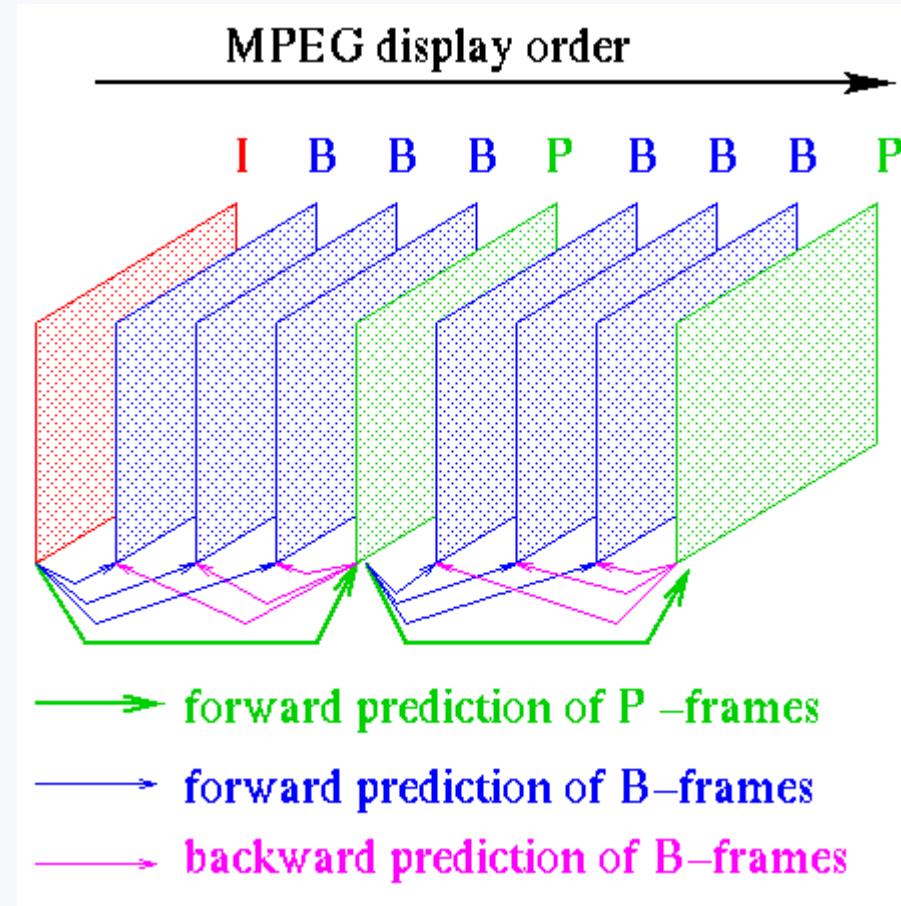
Caracteristici – Compresia

- *P-frame*
 - o imagine codificată predictivă care stochează doar date pentru pixeli diferenți față de cadrul precedent.
 - Exemplu: într-o înregistrare a unei păsări care zboară pe un cer albastru, numai pixelii afectați de mișcarea păsării vor fi înregistrați în P-frame.

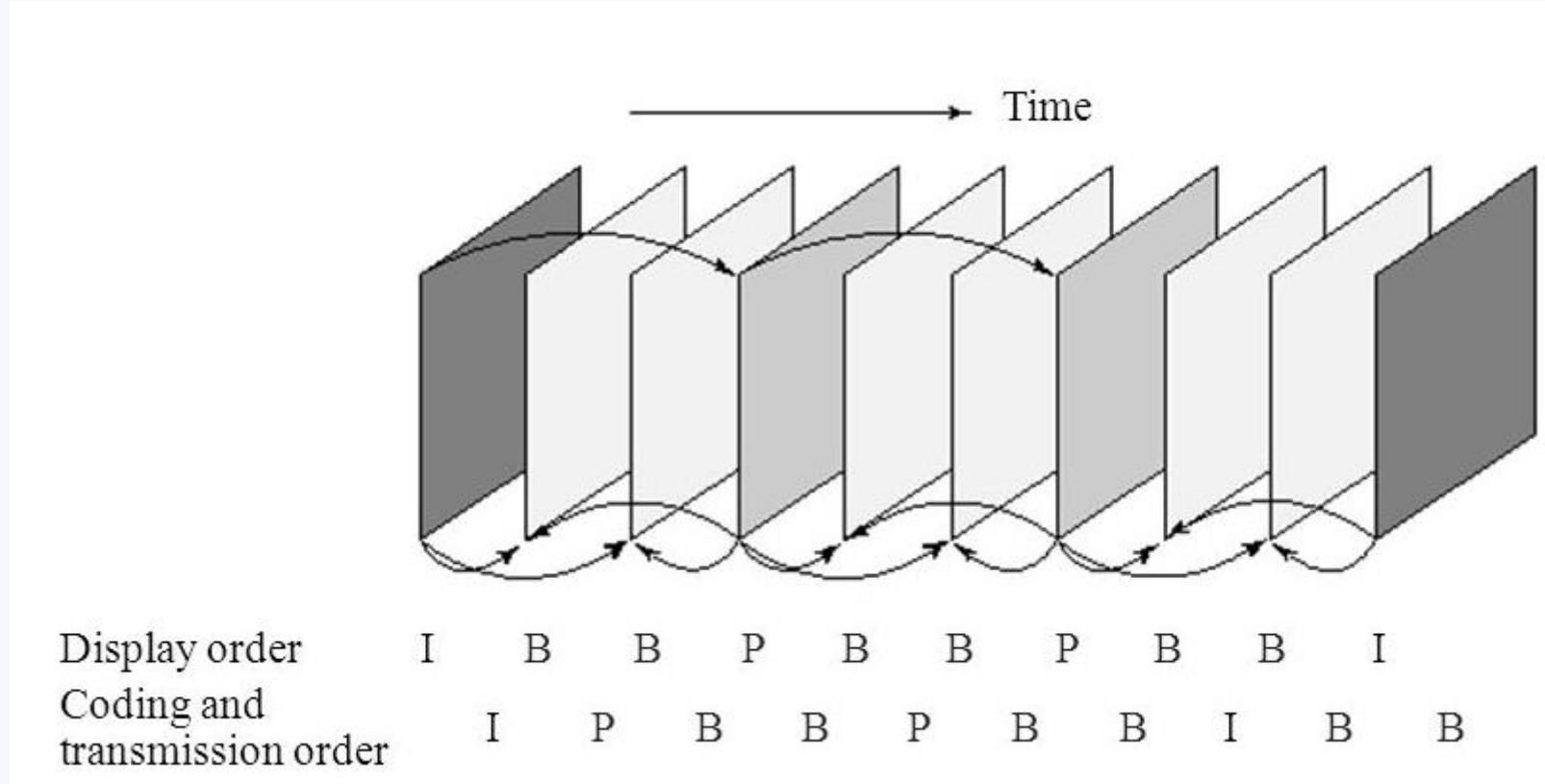
Caracteristici – Compresia

- *B-frame*
 - o imagine codificată predictivă bidirecțională. Înregistrează modificările atât față de cadrul precedent, cât și de cel imediat următor.
 - În medie, un cadru B-frame poate fi comprimat de două ori mai mult decât un cadru P-frame.
- În cazul **compresiei interframe**, cadrele nu sunt comprimate la fel de mult. Cadrele P-frame și cadrele B-frame sunt mai puternic comprimate și necesită mai multă putere de procesare pentru codificare și decodare.

Caratteristici – Compresia



Caratteristici – Compresia



Compresia video



Why Snow and Confetti Ruin YouTube Video Quality (video recomandat de unul dintre colegii voștri)

Caracteristici – Formate Video- Containere

- Container:
 - specifică structura de stocare a componentelor video (imagine + audio) și a datelor asociate (metadate , subtitrări,..)

Caracteristici – Formate Video- Containere

- Matroska Multimedia Container MKV:
 - extensii: .mkv .mk3d .mka .mks
 - formatul *.webm propus de Google are la bază acest format
 - format deschis
 - acceptă aproape orice format audio sau video, ceea ce îl face adaptabil, eficient și considerat unul dintre cele mai bune moduri de stocare a fișierelor audio și video.
 - acceptă mai multe fișiere audio, video și subtitrare, chiar dacă acestea sunt codificate în formate diferite.
 - datorită opțiunilor oferite, precum și gestionării erorilor (posibilitate de redare a fișierelor corupte), a devenit rapid unul dintre cele mai bune containere disponibile în prezent.

Caracteristici – Formate Video- Containere

- **MPEG-4 (MP4):**
 - extensie: .mp4
 - dezvoltat de către Motion Pictures Expert Group și utilizat inițial de QuickTime
 - cel mai frecvent utilizat tip de container pentru a stoca video și audio
 - utilizează codificarea MPEG-4 sau H.264, precum și AAC sau AC3 pentru audio.
 - este acceptat pe scară largă pe majoritatea dispozitivelor de consum și cel mai comun container utilizat pentru videoclipuri online

Caracteristici – Formate Video- Containere

- Alte containere video populare:
 - Ogg - format container gratuit, deschis
 - AVI – Windows
 - MOV – Mac everything
 - MPEG or MPG – creat de MPEG group
 - FLV – Flash video
 - MP4 – creat de MPEG group
 - VOB – DVDs
- Comparatie:
 - https://en.wikipedia.org/wiki/Comparison_of_video_container_formats

Web Video

"Flash was created during the PC era – for PCs and mice. Flash is a successful business for Adobe, and we can understand why they want to push it beyond PCs. But the mobile era is about low power devices, touch interfaces and open web standards – all areas where Flash falls short. The avalanche of media outlets offering their content for Apple's mobile devices demonstrates that Flash is no longer necessary to watch video or consume any kind of web content."

Steve Jobs

Web Video - The Evolution Of Flash



Web Video

- <video> - Elementul HTML utilizat pentru a încorpora conținut video într-un document. Elementul video conține una sau mai multe surse video. Pentru a specifica o sursă video, se va utiliza fie atributul **src**, fie elemente de tip <source> dintre care browser-ul va alege formatul cel mai potrivit.

```
<video controls>
  <source src="foo.webm" type="video/webm">
  <source src="foo.ogg" type="video/ogg">
  <source src="foo.mov" type="video/quicktime">
  I'm sorry; your browser doesn't support HTML5 video.
</video>
```

Web Video

- <video>
 - attribute: *autoplay, controls, src, volume, ...*
 - API: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video>
- HTMLVideoElement (JavaScript)
 - properties: currentSrc, currentTime, duration, ended, error, paused, readyState, volume
 - methods canPlayType, load, pause, play
 - events: canplay, ended, pause, play, volumechange, waiting
- API: <https://developer.mozilla.org/en-US/docs/Web/API/HTMLVideoElement>

Web Video

- <track>
 - HTML element used as a child of the media elements - <audio> and <video>. It lets you specify timed text tracks (or time-based data), for example to automatically handle subtitles.
 - The tracks are formatted in WebVTT format (.vtt files) - Web Video Text Tracks.
 - Link: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/track>

Web Video

- <source>
 - HTML element is used to specify multiple media resources for either the <picture>, the <audio> or the <video> element.
 - It is an empty element.
 - It is commonly used to serve the same media content in multiple formats supported by different browsers..
 - Link: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/source>

```
<video controls>
  <source src="foo.webm" type="video/webm">
  <source src="foo.ogg" type="video/ogg">
  I'm sorry; your browser doesn't support HTML5 video.
</video>
```

Web Video

- recommended video formats:
 - webm – type = video/webm
 - mp4 – type = video/mp4
 - .ogg – type = video/ogg

```
<video controls>
  <source src="foo.webm" type="video/webm">
  <source src="foo.ogg" type="video/ogg">
I'm sorry; your browser doesn't support HTML5 video.
</video>
```

Web Video

- Live Demo: <https://www.w3.org/2010/05/video/mediaevents.html>
- API: <https://www.w3.org/TR/html5/embedded-content-0.html#mediaevents>
- Media Player:
 - [https://developer.mozilla.org/en-US/Apps/Fundamentals/Audio and video delivery/Adding captions and subtitles to HTML5 video](https://developer.mozilla.org/en-US/Apps/Fundamentals/Audio_and_video_delivery/Adding_captions_and_subtitles_to_HTML5_video)

Web Video

```
var W = canvas.width = video.clientWidth;
var H = canvas.height = video.clientHeight;

context.drawImage(video, 0, 0, W, H);
var imageData = context.getImageData(0, 0, W, H);

for (var y = 0; y < H; y++) {
    for (var x = 0; x < W; x++) {
        var i = (y * W * 4) + x * 4;
            // change values in imageData.data[i+...]
    }
}
context.putImageData(imageData, 0, 0);
// other canvas drawing operations
```

Web Video

```
//create the video element using jQuery
var v = $("<video></video>")
    .attr({
        "controls": "",
        "autoplay": "",
        "src": "media/movie.mp4"
    })
    .load()
    .appendTo($("#body"));

// change the video file
v[0].src = "media/test.mp4";
v[0].load();
v[0].play();
```

Web Video – Simple playlist

```
$ (function () {
    var lista = ["movie.mp4", "v2.mp4"];
    var index = 0;

    var video = $( "#myVideo" );

    video.on("ended", function () {
        index = index + 1;
        if (index >= lista.length) index = 0;

        video[0].src = lista[index];
        video[0].load();
        video[0].play();
    });
});
```

Lab Examples

- <https://ase-multimedia.azurewebsites.net/video-player/>
- <https://ase-multimedia.azurewebsites.net/video-processing/>
- <https://ase-multimedia.azurewebsites.net/video-effects/>



Animation

Animația

- Animation is the process of creating the illusion of motion and the illusion of change
- Main techniques:
 - movie technique
 - key frames
 - color changing

Digital Animation

- Digital animation takes two different forms:
- two-dimensional - has evolved from traditional techniques, particularly cel animation;
- three-dimensional - exploits the capabilities unique to the computer to produce an entirely new form of animation.

Digital Animation

- 1. frame-by-frame animation
 - animators produce each successive frame manually
 - technique that provides complete control over frame content, but is also very time consuming

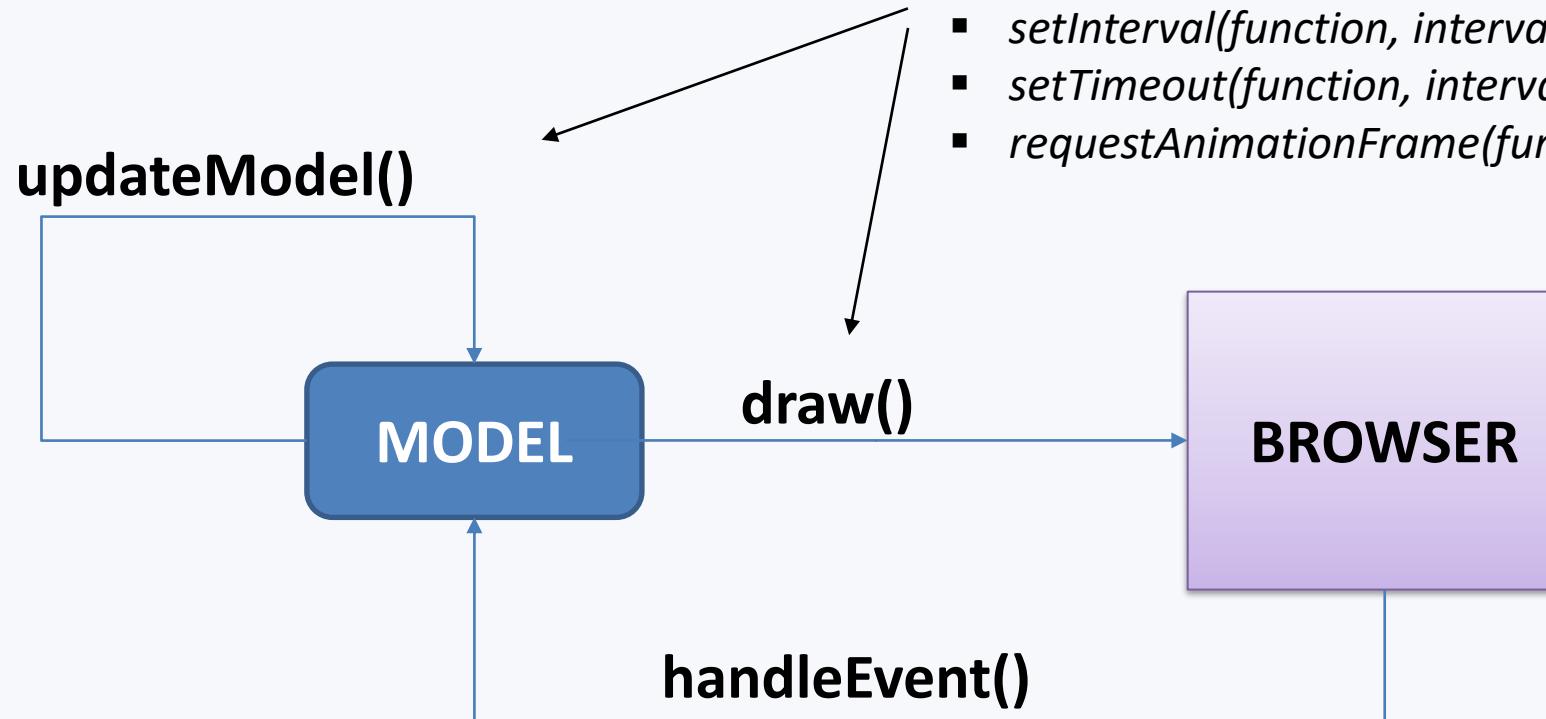
Digital Animation

- 2. tween animation
 - the animator creates the key frames and the computer automatically produces the tweens.
 - There are several different types of tweens:
 - motion tween - can be used to move an object from one position to another. The first key frame places the object in one position and the second places it in another. The program then fills in the intervening frames.
 - path-based animation: the animator draws a path from one key frame to another and the computer fills in the intervening images, spaced out along the path, to produce the desired motion
 - morphing - the shape of one image is gradually modified until it changes into another shape

Digital Animation

- size tweening - the first key frame is the object at its initial size and the second is the final size.
- alpha tweening- color and transparency can be animated using key frames representing initial and final image properties.
- 3. programmed animation

Web Animation



Demo

- lab animation examples



CSS Transitions

- Guide: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions

CSS Animations

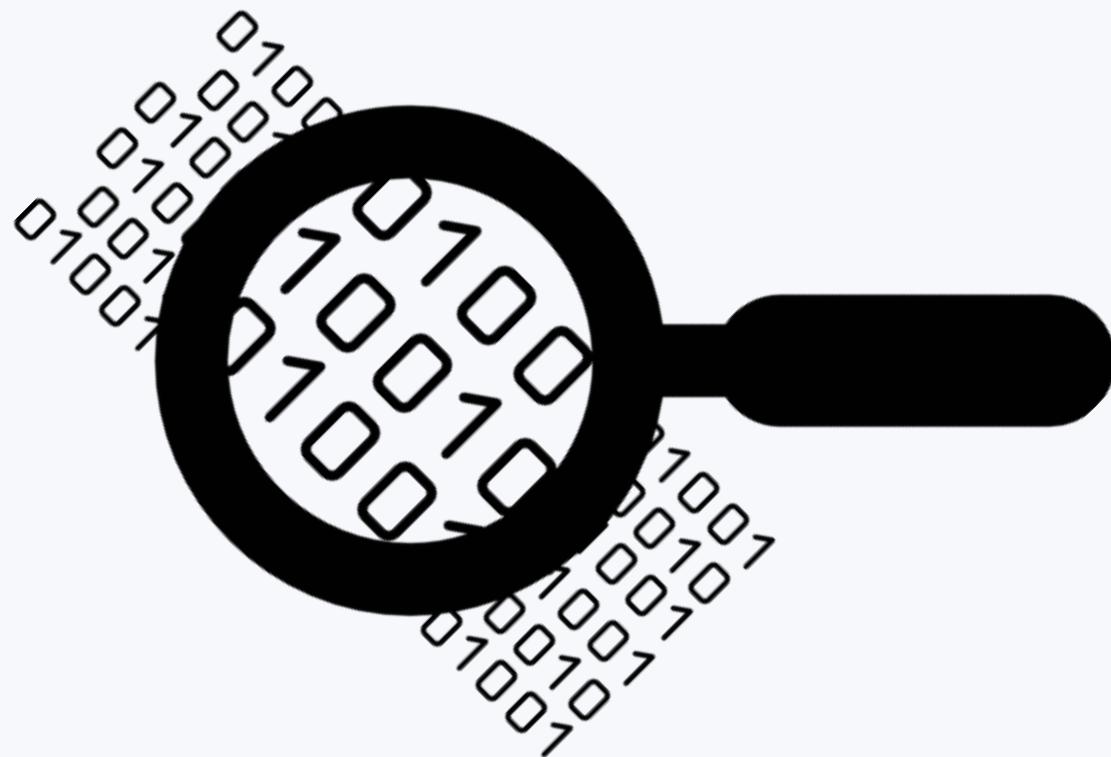
- API: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations
- Guide: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations

Web Animations API

- The **Web Animations API** allows for synchronizing and timing changes to the presentation of a Web page, i.e. animation of DOM elements.
- Limited browser support: <https://caniuse.com/#feat=web-animation> (compared to **CSS Animations** with <https://caniuse.com/#feat=css-animation>)
- API: https://developer.mozilla.org/en-US/docs/Web/API/Web_Animations_API

Demo

<https://css-tricks.com/css-animations-vs-web-animations-api/>



Progressive Web Apps

Progressive Web Apps

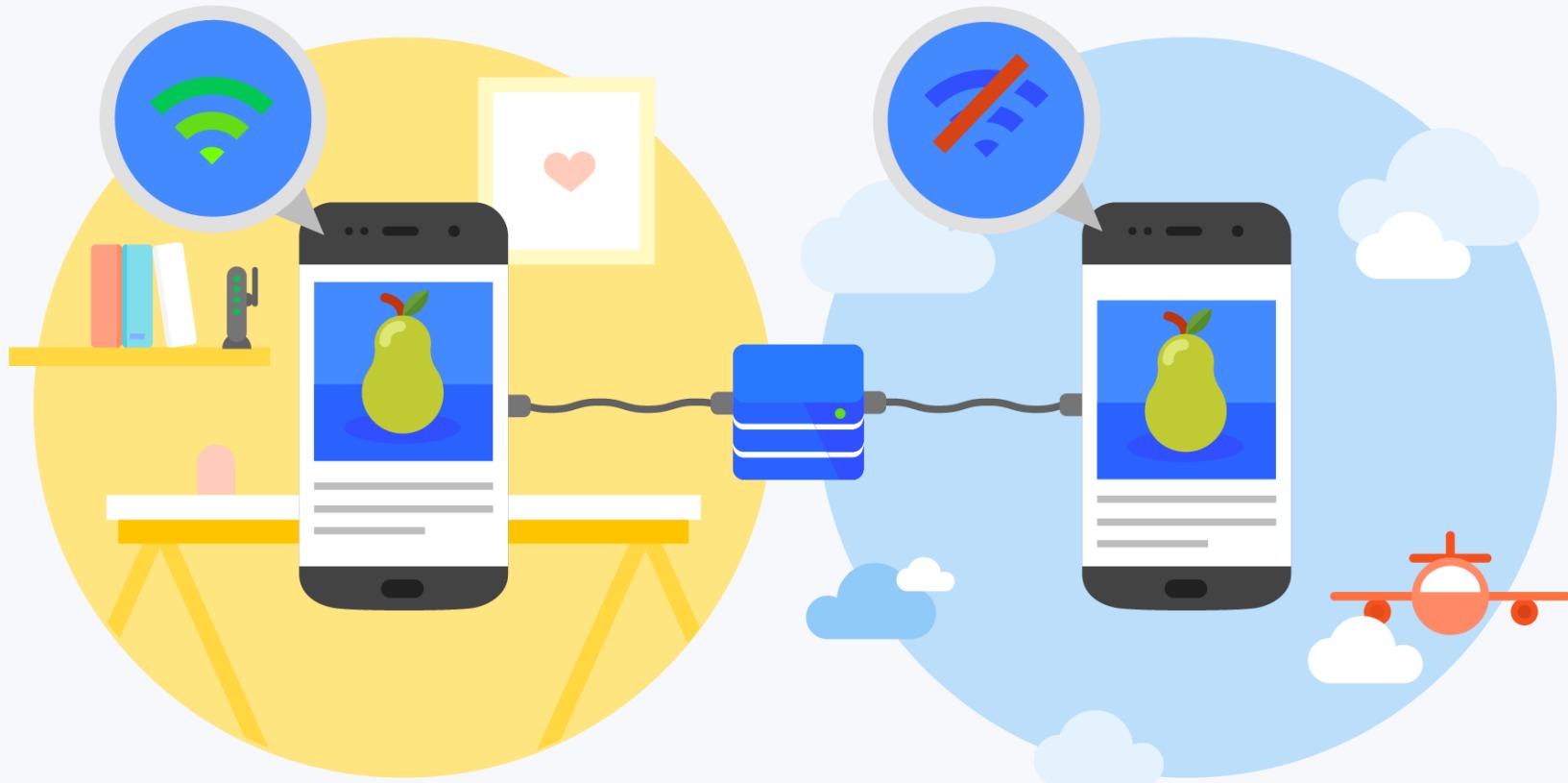
- An application developed using standard web technologies (HTML5, JavaScript, CSS3) which is:
- **Reliable** - load instantly even in uncertain network conditions.
- **Fast** - respond quickly to user interactions with silky smooth animations and no janky scrolling.
- **Engaging** - feel like a natural app on the device, with an immersive user experience.

Reliable

- Progressive web apps are not dependent on the user's connection like traditional websites are.
- When a user visits a progressive web app, it will register a **service worker** that can detect and react to changes in the user's connection. It can provide a fully featured user experience for users who are offline, online, or suffering from an unreliable connection.
- A **service worker**, written in JavaScript, is like a **client-side proxy** and puts you in control of the cache and how to respond to resource requests. By pre-caching key resources you can eliminate the dependence on the network, ensuring an instant and reliable experience for your users.

Reliable

- When launched from the user's home screen, **service workers** enable a Progressive Web App to load instantly, regardless of the network state.



Reliable

- A service worker, written in JavaScript, is like a client-side proxy and puts you in control of the cache and how to respond to resource requests. By pre-caching key resources you can eliminate the dependence on the network, ensuring an instant and reliable experience for your users.

Fast

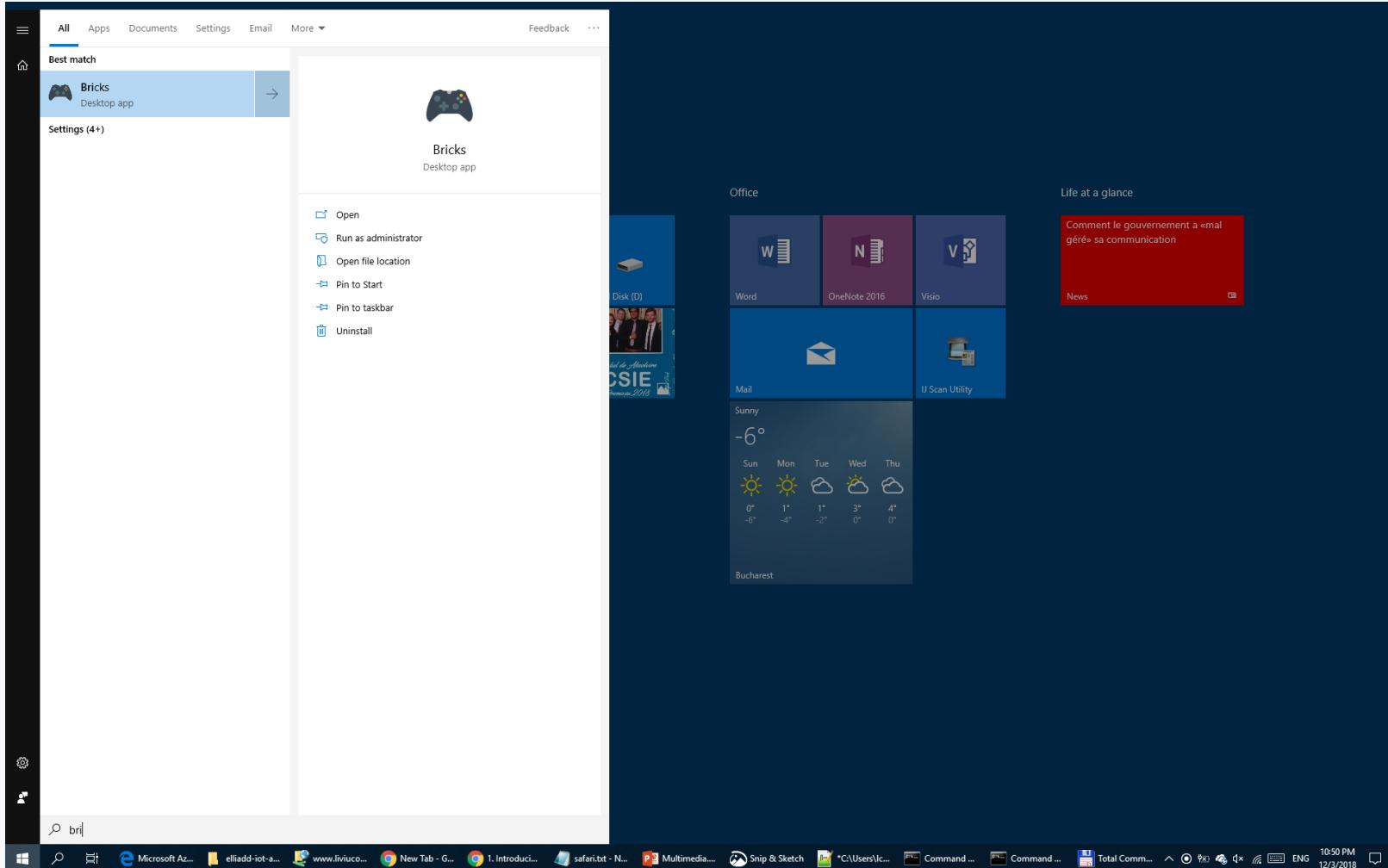
- Using service workers, progressive web apps can launch in an instant, whether the user has a blazing fast connection, an unreliable 2G connection, or even no connection at all. Sites can load in milliseconds, much faster than anything we have experienced on the web before, and sometimes faster than native apps.

Engaging

- Push notifications
 - Progressive web apps can send notifications to their users. The notifications have a completely native feel and are indistinguishable from native app notifications.
- Homescreen shortcut
 - Once a user has shown interest in a progressive web app, the browser will automatically suggest that he add a shortcut to his homescreen—completely indistinguishable from any native app.

Progressive Web Apps

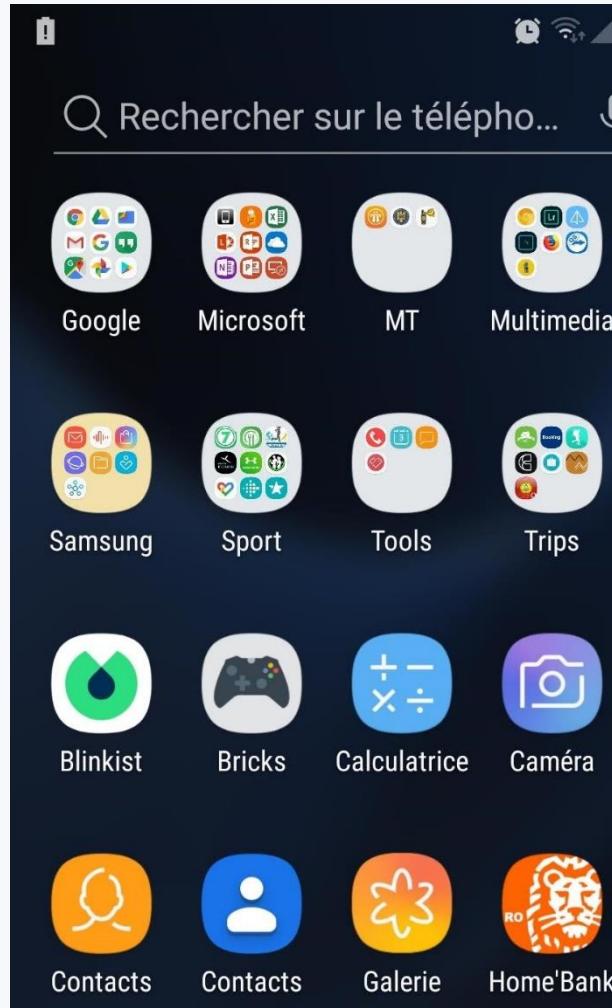
Engaging



Desktop - Homescreen shortcut

Progressive Web Apps

Engaging



Mobile - Homescreen shortcut

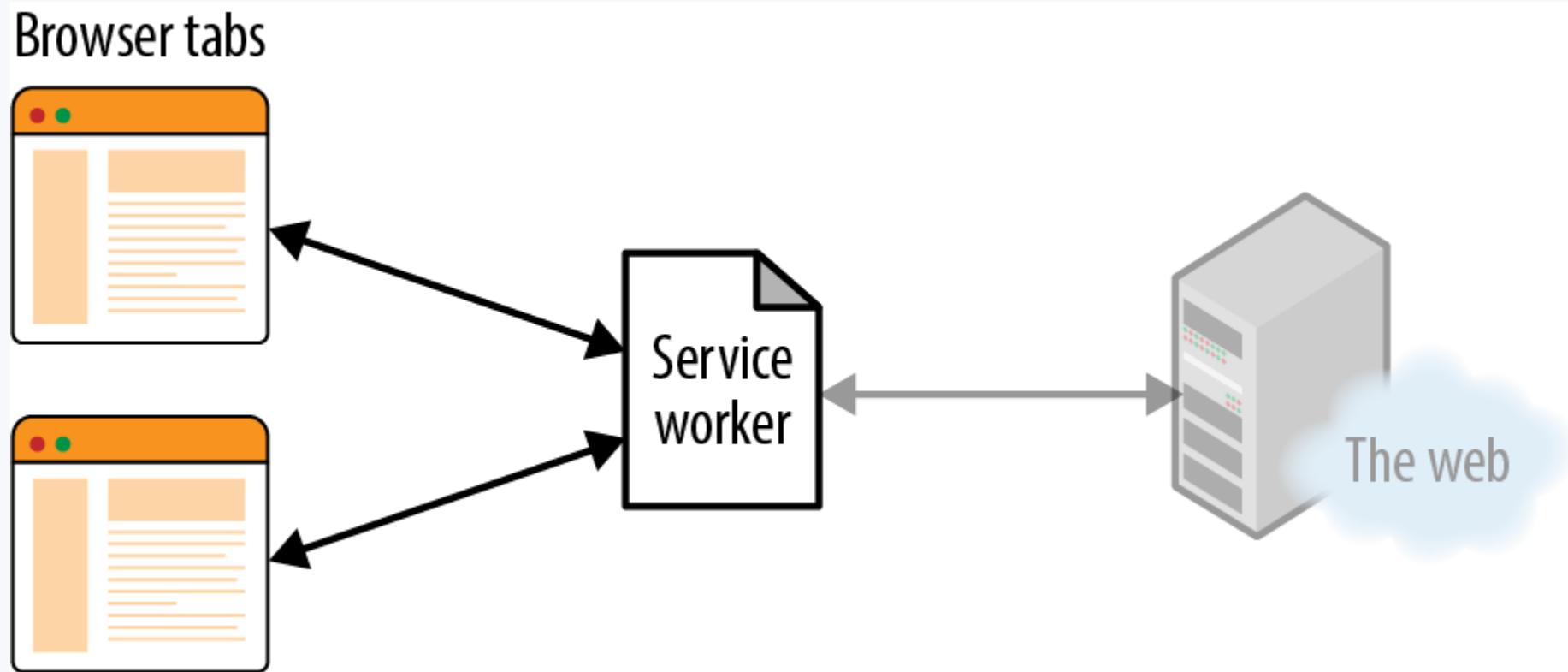
Engaging

- Native look
 - Progressive web apps launched from the homescreen can have an entirely native, app-like look. They can have a splash screen as they are loading. They can launch in full-screen mode, without the browser and phone UI around them. They can even lock themselves to a specific screen orientation (a vital requirement for games).

Service Worker

- At the heart of every progressive web app is the service worker.
- Before service workers, we had code running either on the server or in the browser window. Service workers introduce another layer.
- A service worker is a script that can be registered to control one or more pages of your site. Once installed, a service worker sits outside of any single browser window or tab.
- From this place, a service worker can listen and act on events from all pages under its control. Events such as requests for files from the web can be intercepted, modified, passed on, and returned to the page

Service Worker



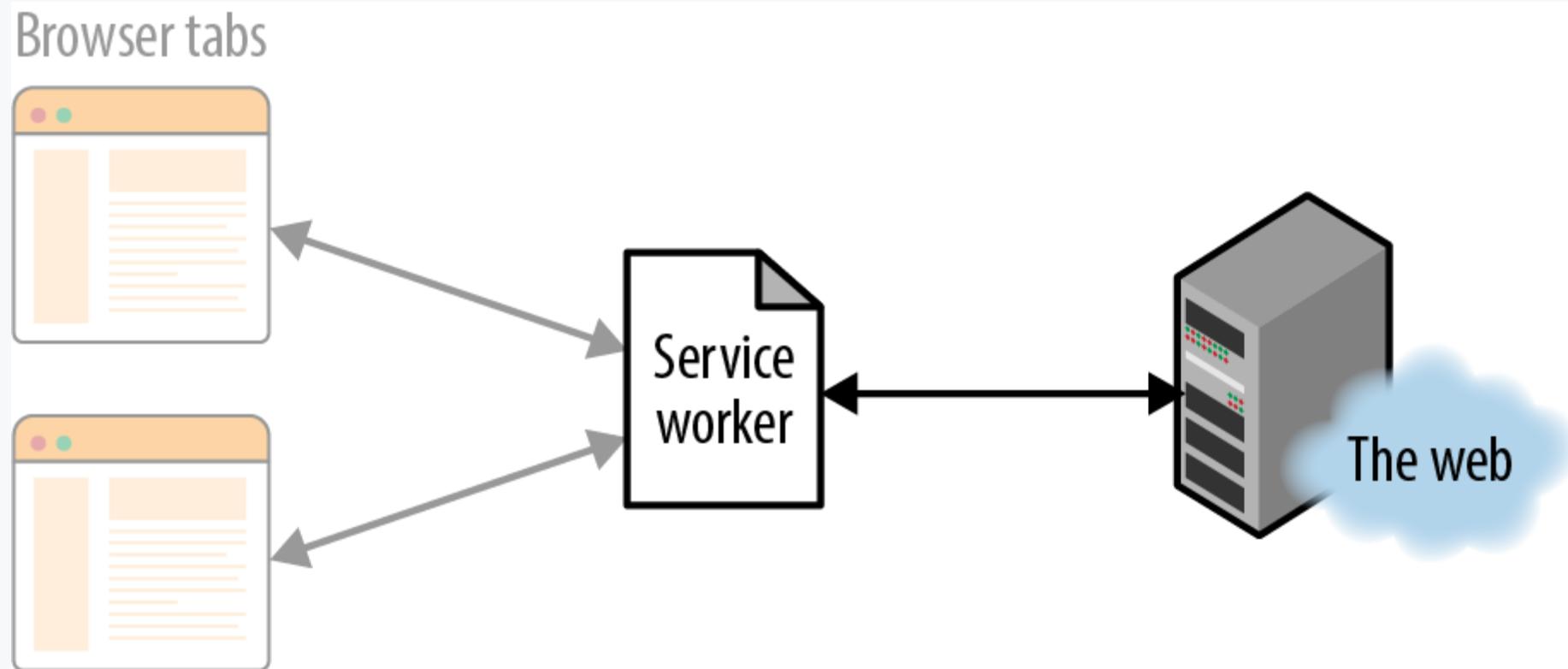
Pages communicating with a service worker while the user is **offline**

Checking the tabs associated with a service worker

- Using the Developer Tools in Google Chrome we can easily check the browser tabs corresponding to a service worker

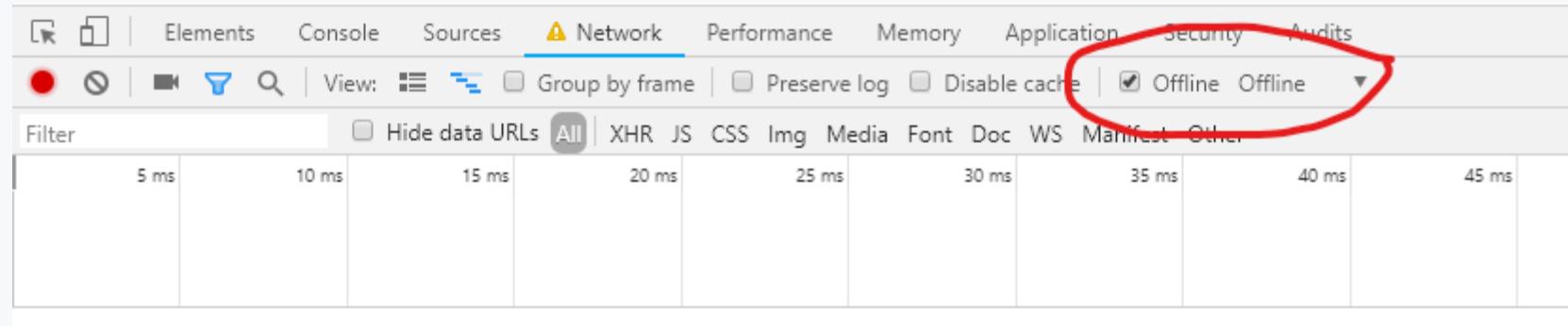
The screenshot shows the 'Service Workers' panel in the Google Chrome DevTools. At the top, there are three checkboxes: 'Offline', 'Update on reload', and 'Bypass for network'. Below this, the address bar shows '127.0.0.1'. Underneath, it says 'Source [serviceworker.js](#) ✖ 8'. It indicates the service worker was 'Received 12/11/2018, 11:40:44 AM'. The status section shows '#5335 activated and is running' with a green dot and a 'stop' link, and '#5501 waiting to activate' with an orange dot and a 'skipWaiting' link. This entry was 'Received 12/17/2018, 10:49:51 PM'. The 'Clients' section lists three clients, each with a yellow background and a 'focus' link: 'http://127.0.0.1:5500/index.html' (repeated twice).

Service Worker



The service worker communicating with a server after a user has left the page

Simulating an offline state in Google Chrome



Service Worker

- Documentation:
 - [https://developer.mozilla.org/en-US/docs/Web/API/Service Worker API](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API)
- A service worker is run in a worker context: it therefore has no DOM access, and runs on a different thread to the main JavaScript that powers your app, so it is not blocking.
- It is designed to be fully async; as a consequence, APIs such as synchronous [XMLHttpRequest](#) and [Web Storage API](#) can't be used inside a service worker.
- Service workers only run over HTTPS, for security reasons.

Service Worker

- Registration:
 - A service worker is first registered using the `ServiceWorkerContainer.register()` method. If successful, the service worker will be downloaded to the client and attempt installation/activation for URLs accessed by the user inside the whole origin, or inside a subset specified by you.
- Documentation:
 - <https://developer.mozilla.org/en-US/docs/Web/API/ServiceWorkerContainer/register>

Service Worker

- Note: check the [documentation](#) for an example also declaring a scope

```
if ("serviceWorker" in navigator) {
  navigator.serviceWorker.register("/serviceworker.js")
    .then(function(registration) {
      console.log("Service Worker registered with scope:", registration.scope);
    }).catch(function(err) {
      console.log("Service worker registration failed:", err);
    });
}
```

Demo

- Create an empty html page and try to register a service worker
 - Note: the registration will fail because we have not yet created the "serviceworker.js" file

```
Service worker registration failed: TypeError: Failed to register a index.html:20
ServiceWorker: A bad HTTP response code (404) was received when fetching the script.
```



Service Worker

- Verify that the current browser supports service workers

```
if ("serviceWorker" in navigator) {
```

- The register call returns a *promise*. If the promise is fulfilled, meaning the service worker was registered successfully, the function defined in the then statement is called. If there was any problem, the function defined inside the catch block would be executed.

```
navigator.serviceWorker.register("/serviceworker.js")
```

- If there was any problem, the function defined inside the catch block would be executed.

Service Worker

- Intercept requests from the Service Worker

```
self.addEventListener("fetch", function(event) {  
    console.log("Fetch request for:",  
    event.request.url);  
});
```

- The code adds an event listener to our service worker by calling addEventListener on self (self within a service worker refers to the service worker itself).
- The function we define to handle these events accesses the request object (available as a property of the fetch event) and logs the URL of that request.

Demo

- Create a service worker and subscribe to the “fetch” event



Service Worker

- every request made by our page (including to third-party servers) now passes through our service worker. All of those requests can now be intercepted, analyzed, and even manipulated. ☺

```
self.addEventListener("fetch", function(event) {  
  if (event.request.url.includes("bootstrap.min.css")) {  
    event.respondWith(  
      new Response(  
        "div {background: green;}",  
        { headers: { "Content-Type": "text/css" } }  
      )  
    );  
  }  
});
```

Service Worker

- Respond to requests with content from the web:

```
self.addEventListener("fetch", function(event) {  
  if (event.request.url.includes("/img/logo.png")) {  
    event.respondWith(  
      fetch("/img/logo-rotated.png")  
    );  
  }  
});
```

Fetch API

- Documentation: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- Provides an interface for fetching resources (including across the network).
- Similar to **XMLHttpRequest**, but the new API provides a more powerful and flexible feature set.
- Moreover, **XMLHttpRequest** is **not** available in ServiceWorkers.

Fetch API

```
// Fetch by URL  
fetch("/img/logo.png");  
// Fetch by the URL in the request object  
fetch(event.request.url);  
// Fetch by passing a request object.  
// In addition to the URL, the request object may contain additional  
headers,  
// form data, etc.  
fetch(event.request);
```

- The first argument in fetch is mandatory and can contain either a request object, or a string with a relative or absolute URL

Fetch API

- The second argument is **optional** and can contain an object with options for the request.

```
return fetch(url, { // Default options are marked with *
  method: "POST", // *GET, POST, PUT, DELETE, etc.
  mode: "cors", // no-cors, cors, *same-origin
  cache: "no-cache", // *default, no-cache, reload, force-cache, only-if-cached
  credentials: "same-origin", // include, *same-origin, omit
  headers: {
    "Content-Type": "application/json; charset=utf-8",
    // "Content-Type": "application/x-www-form-urlencoded",
  },
  redirect: "follow", // manual, *follow, error
  referrer: "no-referrer", // no-referrer, *client
  body: JSON.stringify(data), // body data type must match "Content-Type" header
})
.then(response => response.json()); // parses response to JSON
```

Service Worker - Capturing Offline Requests

- Handle the communication error using "catch"
- Simple "text" response:

```
self.addEventListener("fetch", function(event) {  
  event.respondWith(  
    fetch(event.request).catch(function() {  
      return new Response(  
        "There seems to be a problem with your connection.\n"  
      );  
    })  
  );  
});
```

Service Worker - Capturing Offline Requests

```
var responseContent =  
"<html><body>" +  
"<style>body {text-align: center; background-color: #333; color: #eee; }" +  
"</style>" +  
"<h1>Progressive Web Apps</h1>" +  
"<p>There seems to be a problem with your connection.</p>" +  
"</body></html>;  
  
self.addEventListener("fetch", function(event) {  
  event.respondWith(  
    fetch(event.request).catch(function() {  
      return new Response(  
        responseContent,  
        {headers: {"Content-Type": "text/html"} }  
      );  
    })  
  );  
});
```

Demo

- Check what happens if we remove the headers

- Note:
 - Most web servers are configured to automatically serve most common file types with the correct headers automatically.
 - When a server sends an HTML file, it constructs a response that contains both the HTML, as well as many headers, including a Content-Type header letting the browser know what to do with the response.
 - Because we are constructing a response from scratch, it is up to us to set the correct headers.



Service Worker

- If we modify the Service Worker, the changes **don't immediately** take effect after refreshing the browser. This is because the **old service worker** is still the *active* one, while your new service worker remains in a *waiting* state until the old one is no longer controlling the page.
- To ease development, we can tell the browser to let new service workers take immediate control of the page. In Chrome, this can be done by opening the Application tab of the developer tools, and under the Service Workers section enabling “Update on reload”. This makes sure that each time we change the service worker and refresh the page, the new service worker will immediately take control of the page.

Progressive Web Apps

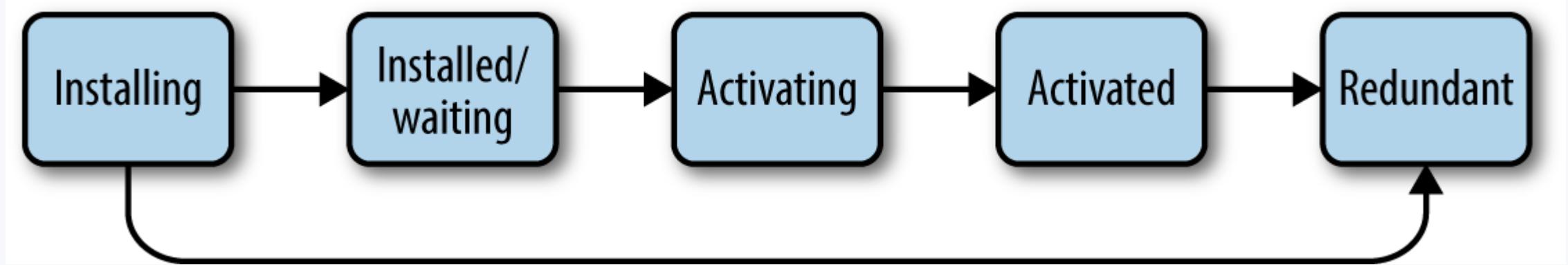
Service Worker

The screenshot shows the Chrome DevTools interface with the Application tab selected. In the left sidebar under 'Application', 'Service Workers' is selected. The main panel displays information about a service worker registered at '127.0.0.1'. The service worker's source is listed as 'serviceworker.js', it was received on '12/4/2018, 9:49:11 AM', and its status is '#4996 activated and is running'. A 'stop' link is available for stopping the worker. Under 'Clients', the URL 'http://127.0.0.1:5500/index.html' is listed with a 'focus' link. At the bottom, there are 'Push' and 'Sync' sections. The 'Push' section contains a text input with 'Test push message from DevTools.' and a 'Push' button. The 'Sync' section contains a text input with 'test-tag-from-devtools' and a 'Sync' button.

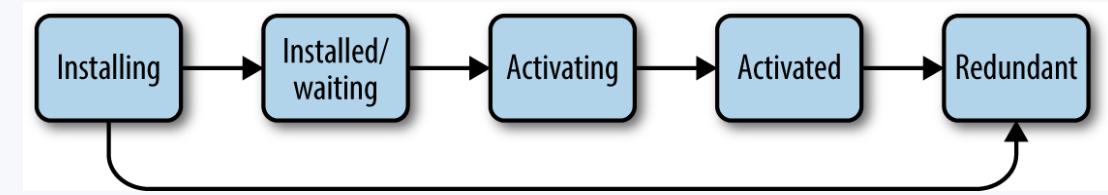
Turning on **Update on reload**

Service Worker LifeCycle

- The lifecycle of a service worker after being downloaded:

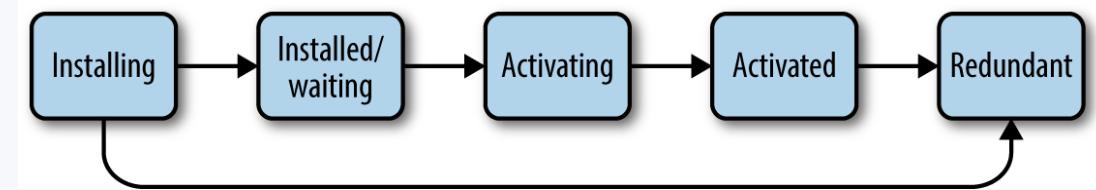


Service Worker LifeCycle



- Download
 - the service worker is immediately downloaded when a user first accesses a service worker–controlled site/page.

Service Worker LifeCycle

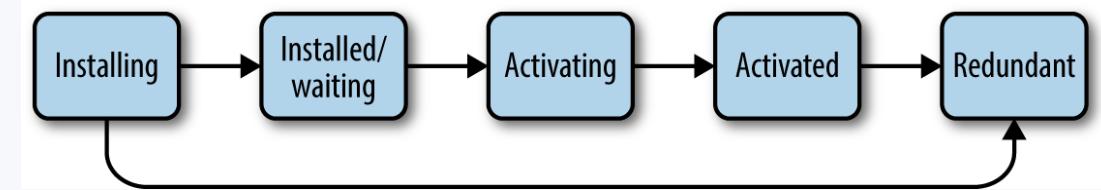


- **Installing state**

- When a new service worker is registered using `navigator.serviceWorker.register`, the JavaScript is downloaded, parsed, and enters the **installing** state.
- If the installation succeeds, the service worker will proceed to the **installed** state.
- The state can be extended by calling `event.waitUntil()` and passing it a promise.

```
self.addEventListener('install', function(e) {  
  e.waitUntil(  
    caches.open(cacheName).then(function(cache) {  
      return cache.addAll(filesToCache);  
    })  
  );  
});
```

Service Worker LifeCycle



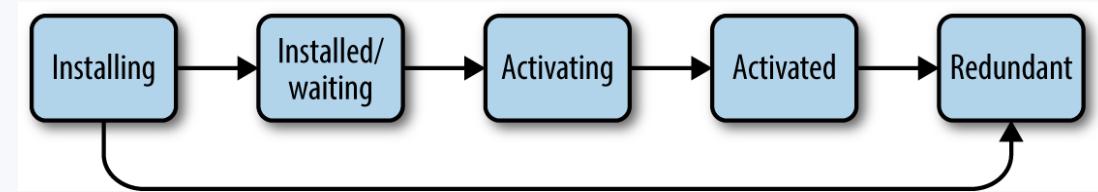
- If an error occurs during installation, the script will instead be moved to the redundant state.

```
self.addEventListener("install", function() {
  console.log("install");
  throw 'error';
});
```

A screenshot of the Chrome DevTools Application tab. The tab bar includes Sources, Network, Performance, Memory, Application (which is underlined), Security, and Audits. Under the Application tab, the 'Service Workers' section is visible. It shows a single entry for '127.0.0.1 - deleted'. The entry details are as follows:

- Source: [serviceworker.js](#) (with a red 'x' icon)
- Received: 12/17/2018, 10:54:46 PM
- Status: ● #5507 is redundant
- Push: Test push message from DevTools. (with a Push button)
- Sync: test-tag-from-devtools (with a Sync button)

Service Worker LifeCycle



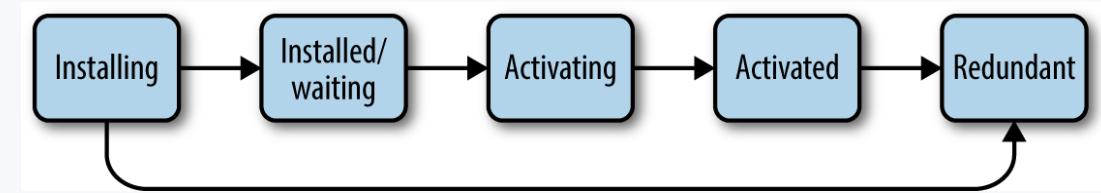
- Installed/waiting state

- Once a service worker has successfully installed, it enters the **installed state**.
- It will then immediately move on to the **activating state** unless another active service worker is currently controlling this app, in which case it will remain **waiting**.

The screenshot shows the Chrome DevTools Application tab for the URL 127.0.0.1. The left sidebar lists 'Manifest', 'Service Workers' (which is selected), and 'Clear storage'. The main panel shows the 'Service Workers' section with the following details:

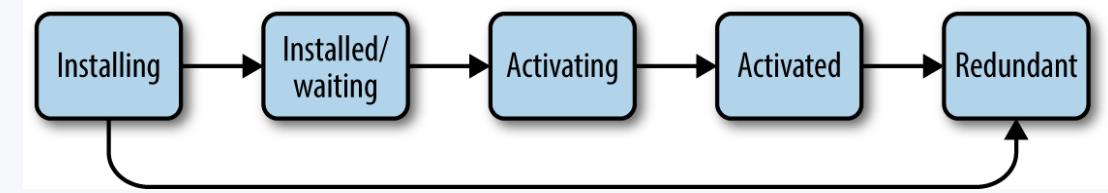
- Source:** [serviceworker.js](#)
- Received:** 12/17/2018, 11:07:59 PM
- Status:** #5509 activated and is running [stop](#)
- #5511** waiting to activate [skipWaiting](#)
- Received:** 12/17/2018, 11:08:15 PM
- Clients:** <http://127.0.0.1:5500/index.html> [focus](#)

Service Worker LifeCycle



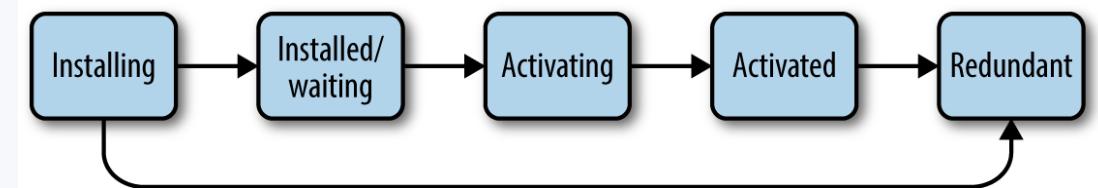
- It is only move to the **activating state** when there are no longer any pages loaded that are still using the old service worker.
- **Why?** Imagine a scenario in which you release a new version of a service worker, and this service worker's install event deletes userdata.json from the cache, adds users.json instead, and changes the fetch event to return the new file when user data is requested. If multiple service workers controlled different pages, the ones controlled by the old service worker might look for the old userdata.json file in the cache after it was removed, causing your app to break.
- Activation can happen sooner using `ServiceWorkerGlobalScope.skipWaiting()` and existing pages can be claimed by the active worker using `Clients.claim()`.

Service Worker LifeCycle



- Activating state
 - Before a service worker becomes active and takes control of the app, the `activate` event is triggered.
 - The **activating state** can be extended by calling `event.waitUntil()` and passing it a promise.

Service Worker LifeCycle

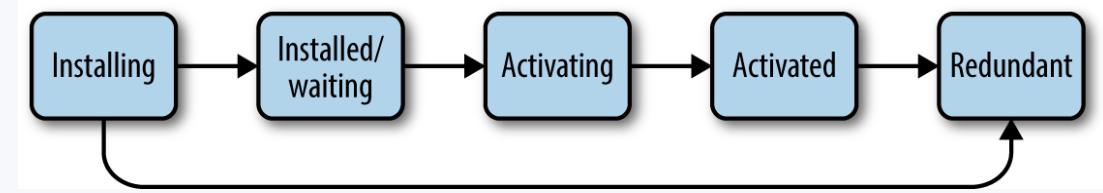


- Activating state

- Before a service worker becomes active and takes control of the app, the `activate` event is triggered.
- The **activating state** can be extended by calling `event.waitUntil()` and passing it a promise.

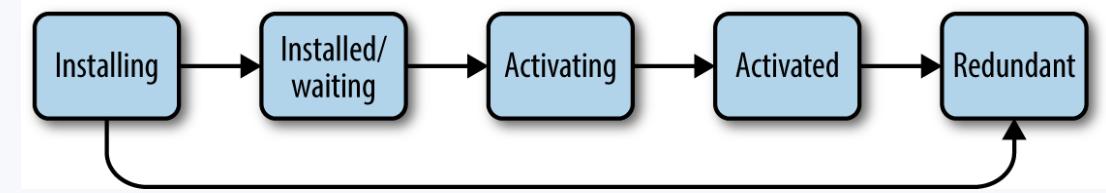
```
self.addEventListener('activate', function(e) {  
  e.waitUntil(  
    caches.keys().then(function(keyList) {  
      return Promise.all(keyList.map(function(key) {  
        if (key !== cacheName) {  
          return caches.delete(key);  
        }  
      }));  
    })  
  );  
  return self.clients.claim();  
});
```

Service Worker LifeCycle



- Activated state
 - Once a service worker is activated, it is ready to take control of the page and listen to functional events (such as `fetch`).
 - Note: a service worker can only take control of pages before they start loading. This means that pages that began loading before the service worker became active cannot be controlled by it.

Service Worker LifeCycle



- Redundant state
 - Service workers that failed during registration, or installation, or were replaced by newer versions, are placed in the **redundant state**. Service workers in this state no longer have any effect on your app.

Service Worker

- Further reading:
 - Service Worker Cookbook (Mozilla) - collection of working, practical examples of using service workers in modern web sites: <https://serviceworke.rs/>

CacheStorage API

- Documentation:
 - <https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/cache-api>
- The **Cache API** was created to enable Service Workers to cache network requests so that they can provide appropriate responses even while offline. However, the API can also be used as a general storage mechanism.

CacheStorage API

- How to check whether the API is available?

```
const cacheAvailable = 'caches' in self;
```

- Creating and opening a cache

```
caches.open('my-cache').then((cache) => {
  // do something with cache...
});
```

CacheStorage API

- Retrieving from a cache using `cache.match`

```
cache.match(request).then((response) => console.log(request, response));
```

- Retrieving all matching responses using `cache.matchAll`.

```
cache.matchAll(request).then((responses) => {
  console.log(`There are ${responses.length} matching responses.`);
});
```

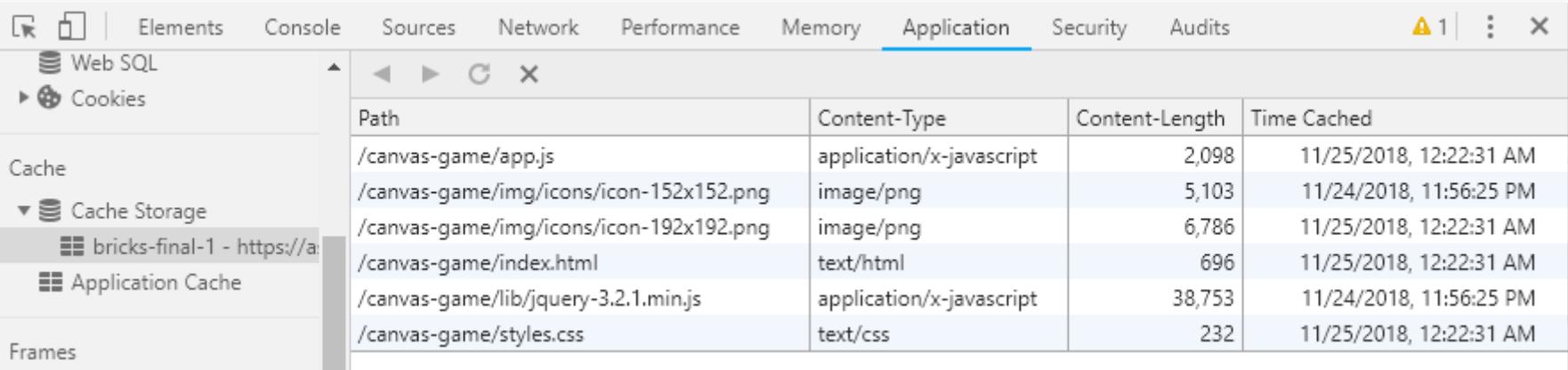
CacheStorage API

- Adding to a cache
 - There are three ways to add an item to a cache - `put`, `add` and `addAll`. All three methods return a Promise.
 - `cache.put`

```
cache.put('/test.json', new Response('{"foo": "bar"}'));
```
 - `cache.add`
 - `cache.addAll`

CacheStorage API

```
let cacheName = 'bricks-final-1';
let filesToCache = [
  'index.html',
  'app.js',
  'styles.css',
  'img/icons/icon-152x152.png',
  'img/icons/icon-192x192.png'
];
```



The screenshot shows the Google Chrome DevTools Application tab. On the left, there's a sidebar with icons for Web SQL, Cookies, Cache, Cache Storage (which is expanded to show 'bricks-final-1 - https://a...', and Application Cache), and Frames. The main area has tabs for Path, Content-Type, Content-Length, and Time Cached. The table lists the cached files from the provided code:

Path	Content-Type	Content-Length	Time Cached
/canvas-game/app.js	application/x-javascript	2,098	11/25/2018, 12:22:31 AM
/canvas-game/img/icons/icon-152x152.png	image/png	5,103	11/24/2018, 11:56:25 PM
/canvas-game/img/icons/icon-192x192.png	image/png	6,786	11/25/2018, 12:22:31 AM
/canvas-game/index.html	text/html	696	11/25/2018, 12:22:31 AM
/canvas-game/lib/jquery-3.2.1.min.js	application/x-javascript	38,753	11/24/2018, 11:56:25 PM
/canvas-game/styles.css	text/css	232	11/25/2018, 12:22:31 AM

Checking the **Cache Storage** in Google Chrome

CacheStorage API

- Deleting an item

```
cache.delete(request);
```

- Where request can be a Request or a URL string.

CacheStorage API

- Deleting a cache

```
caches.delete(key)
```

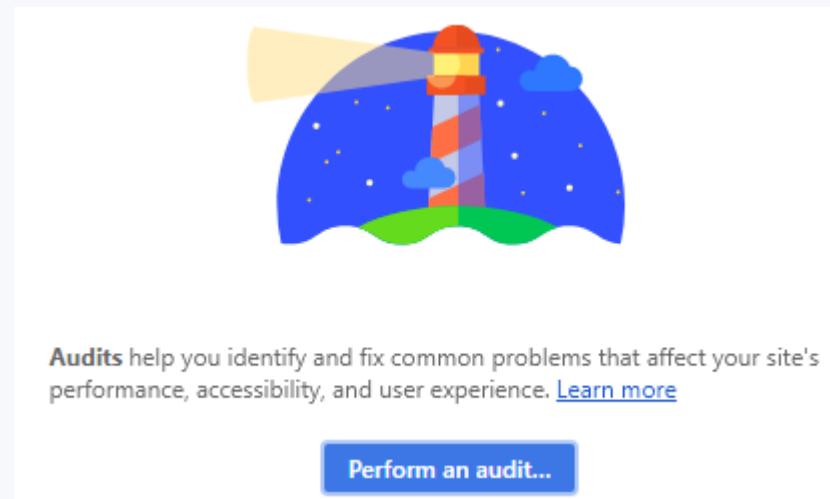
- The function returns a Promise that resolves to true if the cache existed and was deleted, or false otherwise.

CacheStorage API

```
//2. Delete the old version of the cache
self.addEventListener('activate', function(e) {
  e.waitUntil(
    caches.keys().then(function(keyList) {
      return Promise.all(keyList.map(function(key) {
        if (key !== cacheName) {
          return caches.delete(key);
        }
      }));
    })
  );
  return self.clients.claim();
});
```

Lighthouse

- an open-source, automated tool for improving the quality of Progressive Web Apps, that eliminates much of the manual testing that was previously required.
- integrated in Google Chrome.



Is your App a Progressive Web App?

- Checklist: <https://developers.google.com/web/progressive-web-apps/checklist>
- Most of the checks are already included in Lighthouse ☺

Examples



Social Media - Twitter

- Windows Store Application
 - <https://www.microsoft.com/ro-ro/p/twitter/9wzdncrfj140>
- Twitter Lite for Android
 - <https://play.google.com/store/apps/details?id=com.twitter.android.lite&hl=en>

Examples

Drawing- Google Chrome Canvas

- <https://canvas.apps.chrome/>

Web Storage API

Web Storage API

- Documentation:
 - https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API