Bratosin Andrei
323AC

15.01.2024

Tehnologii Web – Tema 3: Image Sharpening cu Canvas API

1. Descrierea aplicației cerute

Cerința presupune realizarea unei aplicații de JavaScript care, în primă fază, obține o imagine de la Dog API și o afișează în browser. Imaginea trebuie să fie afișată împreună cu JSON-ul obținut de la serverul Dog API. Imaginea trebuie apoi prelucrată asincron într-un canvas. Aceasta trebuie mai întâi *inversată* (în cazul meu), și aplicată o temă de procesare. Dintre temele afișate, eu am ales-o pe cea de **Image Sharpening** cu matrice de convoluție. Procesarea trebuie făcută asincron în 4 felii de execuție, cu un decalaj de 1 secundă între felii. Programul mai trebuie să afișeze și durata de execuție pentru fiecare felie în parte.

2. Teoria

2.1. Fetch API

Fetch reprezintă o interfață în JavaScript menită pentru a facilita trimiterea de cereri HTTP și procesarea răspunsurilor. Metoda fetch() a modulului trimite o cerere de GET la destinația dată, iar rezultatul este dat sub forma unui obiect *Response*, care include toate câmpurile răspunsului HTTP.

2.2. Procesarea digitală de imagine

În sistemele digitale, imaginile sunt reprezentate sub formă de matrice, care are pentru fiecare poziție un set de valori RGBA (roșu, verde, albastru, canal alfa). De cele mai multe ori în procesarea de imagine color, această matrice este liniarizată, toate informațiile despre matrice fiind stocate într-un singur vector.

2.3. Procesarea de imagine cu matrice de convoluție

O tehnică foarte utilă în procesarea de imagine s-a dovedit a fi aplicarea unui *filtru de convoluție*: Prin aplicarea unei operații de convoluție de matrice (analoagă cu operația de convoluție de funcții/semnale) între imaginea pentru care se face procesarea și o matrice specială (numită *mască* sau *kernel)* se pot realiza o multitudine de procesări de bază ale unei imagini (blurare, detectare de margini etc.). Natura rezultatului operației de convoluție depinde de forma *kernel*ului ales. Astfel, pentru „ascuțirea” imaginii, am realizat o convoluție între imaginea mea și matricea:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

3. Descrierea implementării

Soluția mea este o aplicație simplă de HTML5. Ca atare, aceasta poate fi rulată direct în browser, fără a fi nevoie de a inițializa vreun server local. De asemenea, folosindu-se doar de JavaScript de bază, nu are nicio dependență externă (în afara comunicării cu Dog API).

Programul folosește următoarele funcționalități ale limbajului JavaScript:

- Canvas API;
- API fetch pentru interacțiunile cu API-uri externe;
- Clasa JSON pentru procesarea și interpretarea răspunsurilor de la API;
- Async/Await pentru operații asincrone;
- Modulul *performance*, care monitorizează timpul de execuție al operațiilor de filtrare;

4. Descrierea funcțională

Latura de procesare a programului este inițializată prin intermediul unui buton. După ce utilizatorul apasă pe buton, au loc următorii pași:

0) Programul verifică mai întâi dacă a mai avut loc vreo execuție. Dacă da, atunci rezultatele execuției anterioare sunt eliminate din DOM;
1) Programul trimite un HTTP GET request către Dog API, de la care primește o poză aleatoare cu un câine, care este afișată utilizatorului printr-un image tag;
    1.1) Dacă conexiunea cu Dog API nu a putut avea loc, utilizatorului îi este doar afișat un mesaj de eroare.
2) Programul generează apoi și un *canvas* în care este inserată imaginea. După transmitere, aceasta este inversată orizontal;
3) Apoi, în 4 felii de execuție, cele 4 cadrane ale imaginii inversate sunt trimise către o funcție care aplică operația de sharpening acestora, iar rezultatul operației este afișat în canvas în locul cadranului original. Între aceste 4 felii de execuție există un decalaj de 1 secundă;
4) Pentru fiecare felie de execuție este stocat timpul de execuție;
5) După ce a fost procesat și actualizat un cadran, timpul de execuție pentru procesarea cadranului este afișat sub canvas.

Această secvență este inițiată și la încărcarea paginii în browser.

5. Modulele aplicației

Aplicația este un site HTML5 vanilla, compus din 3 fișiere:

- Fișierul HTML *index.html*, pagina care este încărcată concret de browser;
- Fișierul de JS *image_processor.js* care conține logica propriu-zisă a aplicației;
- Un fișier de CSS *index.css* în care sunt incluse anumite formatări. Proiectul nu conține elemente de frontend complexe.

Script-ul de JavaScript este compus din 5 funcții:

- *handle_request()* – interacționează cu Dog API și generează div-ul de *wrapper* în care va fi inclusă atât imaginea obținută de la API, cât și canvasul în care se face procesarea imaginii, respectiv div-ul în care se va afișa timpul de execuție al fiecărui cadran în parte;
- *process_image(img_url)* – se ocupă de interacțiunea cu Canvas API, încărcând imaginea în obiectul de canvas, inițializând apoi procesările celor 4 cadrane. În plus, funcția cronometrează aceste 4 operații și le stochează într-o variabilă *timer_list*;
- *apply_filter(image_data, filter_matrix)* – realizează operațiile de convoluție pe matrice în sine;

- *add_timers(timers_list, wrapper)* – funcție care preia obiectul *timer_list* și adaugă timpii memorați în *wrapper;*
- *get_image()* – apelează funcțiile *handle_request* și *process_image* și inserează *wrapper*-ul generat în DOM.

6. Bibliografie

https://dog.ceo/dog-api/documentation/random

https://medium.com/@ramitag18/performing-convolution-on-a-matrix-4682fd364591

https://en.wikipedia.org/wiki/Kernel_(image_processing)#Edge_handling

https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Async_JS

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API