Lekcja 15 – Piksele

Piksele w w obszarze **canvas** rezprezentowane są przez obiekt **ImageData**. Jest to jednowymiarowa tablica, w której każde kolejne 4 komórki reprezentują składowe [R,G,B,A]. Operacji na tym obiekcie możemy dokonywać trzema metodami: **createImageData** – tworzymy pusty obiekt, **getImageData** - pobieramy obiekt z canvas, **putImageData** - utworzony lub pobrany obiekt możemy ponownie umieścić w canvas. Niestety tablica pikseli jest jednowymiarowa i musimy posługiwać się dodatkowymi obliczeniami, aby „dostać się" do każdego piksela i jego składowych kolorów.

Kolejna niedogodność w przekształcaniu grafik wiąże się z zabezpieczeniami stron internetowych przed działaniem hakerów. Metoda **getImageData** w niektórych przeglądarkach jest blokowana – wymagane jest, aby załadowany do obszaru canvas obrazek znajdował się na tym samym serwerze WWW co kod JS. Jedną z metod ominięcia tego problemu podczas pisania i testowania skryptów jest zastosowanie serwera lokalnego, np. XAMPP.

Czerwony kwadrat z pikseli

- **createImageData(100,100)** – rezerwujemy tablicę w pamięci komputera 100x100 pikseli (każdy po 4 bajty) i przypisujemy do zmiennej **czerw**
- w pętli zmienna i „przeskakuje" co 4 bajty
- **data.length** – długość tablicy w bajtach, każda czwórka bajtów (składowe RGBA), to jeden piksel
- **putImageData** – umieszczamy tablicę w obszarze canvas od punktu (0,0) – lewy, górny róg canvas

```
<canvas id="c1" width="800" height="800">
</canvas>
<script>
  var c = c1.getContext('2d')
  var czerw=c.createImageData(100,100);
  for (var i=0;i<czerw.data.length;i+=4){
    czerw.data[i+0]=255;
    czerw.data[i+1]=0;
    czerw.data[i+2]=0;
    czerw.data[i+3]=255;
}
c.putImageData(czerw,0,0);
</script>
```

Losowanie kolorów dla każdego piksela

W poprzednim skrypcie dostęp do pikseli realizowany był w pętli za pomocą jednej zmiennej. Ta wersja umożliwia wybieranie pikseli bardziej intuicyjnie – za pomocą współrzędnych poziomiej X i pionowej Y.

- **(x+y*SZ)*4** – sposób na obliczenie indeksu komórki w tablicy
- losujemy cztery składowe koloru, w tym przeźroczystość

```
var SZ=100;   var WY=150;
var los=c.createImageData(SZ,WY);
for (var y=0;y<WY;y++){
for (var x=0;x<SZ;x++){
  var i=(x+y*SZ)*4;
  for (var p=1;p<4;p++)
    los.data[i+p]=parseInt(Math.random()*255);
}}
c.putImageData(los,100,0);
```

Kolorowy gradient

Gradientowa odmiana poprzedniego skryptu. Do poszczególnych składowych wstawiamy współrzędne. Stosując różnorodne kombinacje arytmetyczne, można uzyskać wiele ciekawych efektów graficznych.

```
var kwa=c.createImageData(256,256);
for (var y=0;y<256;y++){
for (var x=0;x<256;x++){
  var i=(x+y*256)*4;
  kwa.data[i+0]=y;
  kwa.data[i+1]=x;
  kwa.data[i+2]=y-x;
  kwa.data[i+3]=255;
}}
c.putImageData(kwa,0,0);
```

Pobieranie koloru piksela z obrazka

Obrazek jest wyświetlony na stronie WWW i wskazując myszką poszczególne piksele pobieramy składowe kolorów.

- pole komunikatów o identyfikatorze **color**
- **Fkolor** - funkcja pobierająca kolory z piksela wskazywanego przez myszkę
- **layerX** i **layerY** współrzędne myszki
- **getImageData** – w zmiennej **pixel** (prostokąt o wymiarach 1x1) mamy 4 bajty, do których możemy mieć dostęp poprzez właściwość **D**
- w zmiennej **rgba** tworzymy tekstowy opis koloru w postaci ('R,G,B,A')
- **addEventListener** – do zdarzenia **mousemove** przypisujemy funkcję **Fkolor.**

```
<img src=zso.jpg><br>
<div id=color width=200 height=50></div>
<script>
function Fkolor(e) {
  var x=e.layerX;     var y=e.layerY;
  var pixel=c.getImageData(x,y,1,1);
  var D=pixel.data;
  var rgba=
    'rgba('+D[0]+','+D[1]+','+D[2]+','+D[3]+')';
  color.style.background=rgba;
  color.textContent=rgba;
}
document.addEventListener('mousemove', Fkolor);
</script>
```

Filtrowanie (ISLANDIA.JPG)

Typowe filtry wszystkich programów graficznych, to przekształcenia na liczbach, a dokładniej na składowych [R,G,B,A] każdego piksela. W poniższych skryptach pobieramy obraz z obszaru canvas za pomocą **getImageData,** a następnie po przekształceniu go umieszczamy na powrót za pomocą **putImageData**.

Pierwszy skrypt stanowi część bazową, będziemy wstawiać do niego fragmenty odpowiedzialne za poszczególne operacje filtrowania.

```
<canvas id="c1" width="600"
height="1125"></canvas>
<script>
  var c = c1.getContext('2d');
  var obr=new Image();
  obr.src="islandia.jpg";
  obr.onload=function(){
    c.drawImage(obr,0,0);
    var WY=obr.height;
    var SZ=obr.width;
    //FILTRY
    ...
  }
</script>
```

Wybór składowej

Jeśli chcemy zobaczyć, jak wygląda jedna składowa kolorowego obrazu, to wystarczy wyzerować pozostałe dwie. Skrypt w ramce pokazuje składową czerwoną – wyzerowane składowe niebieska i zielona. Intensywność pozostawiamy na tym samym poziomie.

```
var D=c.getImageData(0,0,600,225);
for (var y=0;y<WY;y++){
for (var x=0;x<SZ;x++){
  var i=(x+y*SZ)*4;
  D.data[i+1]=0;
  D.data[i+2]=0;
}}
c.putImageData(D,0,225);
```

Inwersja - negatyw

Barwę każdego piksela zamieniamy na dopełniającą - odejmujemy ją od 255. Intensywność pozostawiamy na tym samym poziomie.

```
var D=c.getImageData(0,0,600,225);
for (var y=0;y<WY;y++){
for (var x=0;x<SZ;x++){
  var i=(x+y*SZ)*4;
  for (var p=0;p<3;p++)
    D.data[i+p]=255-D.data[i+p];
}}
c.putImageData(D,0,450);
```

Desaturacja - odcienie szarości

Jak wygenerować szarości z kolorów? To nieco bardziej skomplikowane zagadnienie związane ze sposobem w jaki działa ludzkie oko. Wrażenie szarości uzyskujemy, gdy na przykład uśrednimy wszystkie trzy składowe. Intensywność pozostawiamy na tym samym poziomie.

```
var D=c.getImageData(0,0,600,225);
for (var y=0;y<WY;y++){
for (var x=0;x<SZ;x++){
  var i=(x+y*SZ)*4;
  var R=D.data[i+0];
  var G=D.data[i+0];
  var B=D.data[i+0];
  var gray=(R+G+B)/3;
  for (p=0;p<3;p++)
    D.data[i+p]=gray;
}}
c.putImageData(D,0,675);
```

Przeźroczystość

Aby obrazek był widoczny w 100%, czwarty bajt piksela należy ustawić na 255, aby był niewidoczny – ustawiamy go na 0. Przeźroczystość z gradientem wymaga jedynie przeliczenia intensywności wyświetlania w zależności od wysokości lub szerokości na której znajduje się badany piksel.

```
var D=c.getImageData(0,0,600,225);
for (var y=0;y<WY;y++){
for (var x=0;x<SZ;x++){
  var i=(x+y*SZ)*4;
  D.data[i+3]=255*y/WY;
  //D.data[i+3]=255*(WY-y)/WY;
}}
c.putImageData(D,0,900);
```