Lekcja 6 – Tablice – Losowanie

Jak sprawić, aby po ekranie „fruwało” 100 kwadratów? Definiować tyle zestawów zmiennych? Z pomocą przychodzą tablice, które są zestawem komórek, w których można zapamiętać wiele zmiennych.

Definicja pustej tablicy i przypisywanie do niej jest niezwykle proste. Pamiętać należy, że pierwszy element tablicy ma numer „zero”. Do każdego elementu tablicy można przypisać liczbę, tekst albo obiekty złożone, np. inną tablicę – w tym przypadku mamy do czynienia z tablicami wielowymiarowymi.

```
var TK = [];
TK[0] = 100;
```

```
var TK = [];
TK[0] = [100, 100, 50, 0, „red”, 1, 2, 1];
var kolor = TK[0][4];
```

W tablicy **TK**, w elemencie o numerze [0] zapamiętano 8 danych koniecznych do animowania jednego kwadratu, w kolejności: **x, y, bok, kąt, kolor, vx, vy, vk.** Aby zapamiętać dane dla 100 kadratów, należy zdefiniować 100 takich zestawów. Najłatwiej oczywiście zrobić to za pomocą losowania.

Losowanie

Funkcja **Math.random** losuje liczby rzeczywiste z przedziału 0..1. Funkcja **Math.round** zaokrągla liczbę rzeczywistą do całkowitej. Przygotowana funkcja **losowa(p,k)** losuje liczby całkowite z przedziału p..k.

W podobny sposób można losować kolory dla funkcji RGB lub RGBA. Zwróć uwagę na połączenie tekstów ze zmiennymi liczbowymi.

```
function losowa(p,k) {
  return Math.round(Math.random()*(k-p)+p);
}
```

```
function LosRGBA(){
  var r = losowa(0,255);
  var g = losowa(0,255);
  var b = losowa(0,255);
  var a = Math.random();
  return "rgb("+r+","+g+","+b+","+a+")";
}
```

Losowanie zmiennych początkowych

Wykorzystujemy funkcje **losowa** i **LosRGBA** do przygotowania zestawy zmiennych, które wstawiamy do kolejnych komórek tablicy TK

```
var ILE = 100;
for (var i = 0; i < ILE; i = i + 1){
  var x = losowa(100,300);
  var y = losowa(100,300);
  var bok = losowa(1,100);
  var kat = losowa(0,45);
  var kolor = LosRGBA();
  var vx = losowa(-5,5);
  var vy = losowa(-5,5);
  var vk = losowa(-5,5);
  TK[i] = [x,y,bok,kat,kolor,vx,vy,vk];
}
```

Rysowanie i odbicia

Za pomocą pętli FOR będziemy rysować wszystkie kwadraty, przesuwać, obracać i sprawdzać odbicia. Wszystkie dotychczasowe zmienne dla pojedynczego kwadratu zamieniamy na odpowiadające im pozycje w tablicy: zamiast **x** wykorzystamy element zerowy tablicy **TK[i][0]**, zamiast **y** – element pierwszy **TK[i][1]**, itd.

```
for (i=0;i<ILE;i=i+1){
  kwaO(TK[i][0],TK[i][1],TK[i][2],TK[i][3],TK[i][4]);

  TK[i][0]=TK[i][0]+TK[i][5];
  TK[i][1]=TK[i][1]+TK[i][6];
  TK[i][3]=TK[i][3]+TK[i][7];
  if (TK[i][0] > w || TK[i][0] < TK[i][2]){
    TK[i][5]=-TK[i][5];
    TK[i][7]=-TK[i][7];
  }
  if (TK[i][1] > h || TK[i][1] < TK[i][2]){
    TK[i][6]=-TK[i][6];
    TK[i][7]=-TK[i][7];
  }
  if (TK[i][3]>=360){
    TK[i][3]=0;
  }
}
```

Obiekt - lista

Czy zwróciłeś uwagę, jak niewygodne jest operować numerami komórek, w których znajdują się konkretne zmienne: x – TK[i][0] … Ten problem da się rozwiązać za pomocą **list**, które zapamiętują dane pod konkretną nazwą podczas gdy tablica zapamiętywała zmienną pod konkretnym numerem.

Lista **Rk** również mieści w sobie zestaw zmiennych potrzebnych do animowania jednego kwadratu. Jednak dostęp do tych danych jest dużo wygodniejszy, bo poprzez konkretną nazwę.

```
var ILE = 100;
for (var i = 0; i < ILE; i = i + 1){
  var Rk={};
  Rk.x=losowa(100,300);
  Rk.y=losowa(100,300);
  Rk.bok=losowa(1,100);
  Rk.kat=losowa(0,45);
  Rk.kolor=LosRGBA();
  Rk.vx=losowa(-5,5);
  Rk.vy=losowa(-5,5);
  Rk.vk=losowa(-5,5);
  TK[i]=Rk;
}
```

Rysowanie i sprawdzanie wygląda bardzo podobnie, ale o wiele łatwiej się czyta program.

```
for (i=0;i<ILE;i=i+1){
  kwaO(TK[i].x,TK[i].y,TK[i].bok,TK[i].kat,TK[i].kolor);
  TK[i].x = TK[i].x + TK[i].vx;
  TK[i].y = TK[i].y + TK[i].vy;
  TK[i].kat = TK[i].kat + TK[i].vk;
  if (TK[i].x > w || TK[i].x < TK[i].bok){
    TK[i].vx = -TK[i].vx;
    TK[i].vk = -TK[i].vk;
  }
  if (TK[i].y > h || TK[i].y < TK[i].bok){
    TK[i].vy = -TK[i].vy;
    TK[i].vk = -TK[i].vk;
  }
  if (TK[i][3]>=360){
    TK[i][3]=0;
  }
}
```

ODCINEK

Z listami już mieliśmy do czynienia podczas rysowania trójkątów. Funkcja ODCINEK zwracała dwie współrzędne w postaci **listy**.

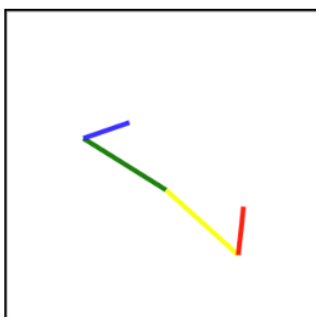
```
function ODCINEK(x,y,bok,kat){
  var x1=x+bok*Math.cos(kat*Math.PI/180);
  var y1=y+bok*Math.sin(kat*Math.PI/180);
  c.beginPath();
  c.moveTo(x,y); c.lineTo(x1,y1);
  c.stroke();
  return {x1,y1};
}
```

Animowany łamaniec

Po ekranie przesuwają się i obracają cztery połączene ze sobą odcinki. Aby obliczeć położenie odcinków na końcach dłuższych ramion wykorzystaliśmy listę, którą zwraca funkcja ODCINEK. Punkt **P1** i **P2**, to listy zawierające współrzędne – one określają początki ramion krótszych.

Zwróć uwagę na sposób ich wykorzystania: współrzędna x zawarta jest na liście **P1.x1** i **P2.x1**, a współrzędne y na liście **P1.y1** i **P2.y1**.

Nie zapomnij zdefiniować brakujących zmiennych.



```
function animacja() {
  c.clearRect(0, 0, w, h);
  c.strokeStyle="black";
  c.strokeRect(0, 0, w, h);
  x=x+vx;  y=y+vy;  kat=kat+1;

  c.lineWidth=5;
  c.strokeStyle="green";
  var P1=ODCINEK(x,y,bok,kat);
  c.strokeStyle="yellow";
  var P2=ODCINEK(x,y,bok,-kat*1.5);
  c.strokeStyle="blue";
  ODCINEK(P1.x1,P1.y1,bok/2,-kat*2+45);
  c.strokeStyle="red";
  ODCINEK(P2.x1,P2.y1,bok/2,kat*3);

  if (x<0 || x>w){vx=-vx}
  if (y<0 || y>h){vy=-vy}
  clearTimeout(czas);
  czas = setTimeout(animacja, skok);
}
animacja();
```