LIVE – Gra w życie

Live jest jednym z pierwszych i najbardziej znanych tzw. automatów komórkowych. Został wymyślony w 1970 roku przez brytyjskiego matematyka Johna Conwaya. Co to takiego automat komórkowy? Mnóstwo sąsiadujących z sobą komórek. Każda z nich może być zapalona lub zgaszona zgodnie z wymyślonymi prostymi regułami. Na czym polega gra – symulacja? Ustalamy układ początkowy komórek, puszczamy cały układ w ruch i obserwujemy. Więcej na Wikipedii: https://pl.wikipedia.org/wiki/Gra_w_życie

LIVE w JavaScript – krok po kroku

ANIMACJA
Rozpoczynamy od podstawowego schematu stosowanego w animacji

```
<canvas id="c1" width="600"
        height="600">
</canvas>

<script>
var c = c1.getContext('2d')

var w = c.canvas.width;
var h = c.canvas.height;
var skok = 10;
var czas;
function animacja() {
  c.clearRect(0, 0, w, h);
  // ramka
  c.strokeStyle="black";
  c.strokeRect(0, 0, w, h);
  //...
  clearTimeout(czas);
  czas = setTimeout(animacja, skok);
}

animacja();
</script>
```

KWADRATY
Plansza składa się z kwadratowych komórek. Lewy górny róg komórki o długości (bok) znajduje się w punkcie (x,y). Brzeg komórki ma kolor (kol) i wnętrze jest wypełnione kolorem (wyp).

```
function KWA(x,y,bok,kol,wyp){
  c.lineWidth=1;
  c.fillStyle=wyp;
  c.fillRect(x,y,bok,bok);
  c.strokeStyle=kol;
  c.strokeRect(x,y,bok,bok);
}
```

Sprawdzenie działania:

```
KWA(10,10,100,"black","white");
```

SZACHOWNICA
Szachownica składa się z kwadratów rysowanych w dwóch pętlach – położenie lewego, górnego rogu obliczane jest prostymi zależnościami: x+bok*k oraz y+bok*w

```
function SZACHY(x,y,bok,ile,kol,wyp){
  for (var k=0; k<ile; k++){
  for (var w=0; w<ile; w++){
    KWA(x+bok*k,y+bok*w,bok,kol,wyp);
  }}
}
```

Sprawdzenie działania:

```
SZACHY(10,10,20,10,"black","silver");
```

SUWAK
Szachownica powinna zmieniać swoje wymiary i automatycznie dopasowywać się do wielkości okna w przeglądarce – zrealizujemy to za pomocą suwaka.

```
<br>
<input type="range" id="SUWile" min=1 max=100 onchange=FSuwak()>
<input type="text" id="SUWtxt" size=6 onchange=FSuwak()>
```

Zakładamy, że szachownica może mieć od 1 do 100 kratek. Odpowiednie instrukcje w HTML i funkcja FSuwak je obsługująca

```
function FSuwak(){
  SZAile = SUWile.value;
  SUWtxt.value = SZAile
  SZAbok = SZAsze / SZAile;
}
```

START

Deklarujemy zmienne globalne SZAsze, SZAile i SZAbok odpowiadające za wielkość szachownicy. Umieścimy je w funkcji startowej START – będzie inicjowała wszystkie parametry.

```
function START(){
  SZAsze=580;
  SZAile=10;
  SZAbok=SZAsze / SZAile;
  SUWile.value=SZAile;
  FSuwak();
}
```

Pobieramy do zmiennej SZAile wartość ustawioną na suwaku
W polu tekstowym piszemy ilość kratek szachownicy
Obliczamy bok szachownicy
W funkcji START wstawiamy początkowe ustawienia suwaka

Funkcję START wstawiamy na koniec, przed funkcję animacja();

```
SZACHY(10,10,SZAbok,SZAile,"black","silver");
```

Teraz wystarczy delikatnie poprawić rysowanie szachownicy w animacji

KLIKANIE

Najpierw nauczymy się sprawdzać, w który punkt obszaru canvas kliknięto myszką, a potem przeliczymy uzyskane współrzędne na numer kolumny i wiersza na szachownicy.

```
var MYSZKA={x:0,y:0};
var SZAWSP={x:0,y:0};
```

Deklarujemy dwie zmienne, w której zapamiętamy:
- MYSZKA współrzędne kliknięcia myszki
- SZAWSP numer wiersza i kolumny szachownicy (przeliczone ze współrzędnych myszki)

Do pobrania współrzędnych użyjemy standardowej funkcji **onclick.** Obszar canvas jest przesunięty o 8 pikseli w przeglądarkach!

Od położenia myszki odejmujemy 10 (o tyle przesunięta jest szachownica), a następnie dzielimy przez długość boku i to wystarczy aby obliczyć współrzędne kratek.

```
c.canvas.onclick =  function(evt) {
  MYSZKA.x = evt.clientX-8;
  MYSZKA.y = evt.clientY-8;
  SZAWSP.x=parseInt((MYSZKA.x - 10) / SZAbok) + 1;
  SZAWSP.y=parseInt((MYSZKA.y - 10) / SZAbok) + 1;
  alert('('+MYSZKA.x+','+MYSZKA.y+')' +
        '('+SZAWSP.x+','+SZAWSP.y+')');
}
```

Aby sprawdzić działanie, użyjemy funkcji **alert**, którą wstawiamy (na chwilę) do funkcji onclick.

WYPEŁNIANIE

Aby wypełnić kliknięta kratkę należy obliczyć współrzędne jej lewego, górnego rogu (na podstawie numeru wiersza i kolumny (na podstawie kliknięgo punktu)) i narysować kwadrat. Odpowiednie polecenia wstawiamy również do funkcji onclick.

```
var x1=(SZAWSP.x-1) * SZAbok + 10;
var y1=(SZAWSP.y-1) * SZAbok + 10;
KWA(x1,y1,SZAbok,"black","white");
```

Aby zobaczyć działanie należy na chwilę wyłączyć animację i ręcznie narysować szachownicę.

```
//animacja();
SZACHY(10,10,SZAbok,SZAile,"black","silver");
</script>
```

TABLICE

Jak sprawdzać kolor kratki? Pobierać z canvas kolor piksela? Tak też można, ale jest to nieefektywne. Na pewno lepsze będą tablice, w których zapamiętamy stan kratek, i które za chwilę będą niezbędne w celu uruchomienia „życia”.

```
function TABLICE(){
  SZAtab=new Array(101);
  SZAsas=new Array(101);
  for (var w=0; w <= 101; w++) {
    SZAtab[w] = new Array(101);
    SZAsas[w] = new Array(101);
    for (var k=0; k <= 10; k++) {
      SZAtab[w][k] = 0;
      SZAsas[w][k] = 0;
    }
  }
}
```

Deklarujemy dwie tablice globalne (bez var) i je zerujemy:
SZAtab – przechowujemy stan kratek: 0-szara, 1-biała
SZAsas – liczba białych sąsiadów wokół kratki

Tablice mają z „każdej strony” dodatkowy wiersz i kolumnę o numerach 0 i 101 – nie są używane, ale będą bardzo użyteczne podczas wyszukiwania i obliczania – nie musimy stosować dodatkowych warunków brzegowych.

Funkcję TABLICE wstawiamy do funkcji START

POCZĄTEK

Gdy klikamy myszą w kratkę powinna zapalić się na biało, gdy klikniemy powtórnie - ma wrócić do stanu początkowego (kolor szary). Aby było to możliwe musimy „zapalać” – wstawiać 1 lub „gasić” – wstawiać 0, do odpowiednich komórek tablicy SZAtab.

Zamiana zawartość klikniętej komórki tablicy na przeciwną.

```
SZAtab[SZAWSP.x][SZAWSP.y] = !(SZAtab[SZAWSP.x][SZAWSP.y]);
```

Nie jest nam już potrzebne zapalanie i gaszenie w funkcji onclick – usuwamy wyliczanie współrzędnych x1 i y1 oraz rysowanie kwadratu – szachownica będzie rysowana na podstawie tablicy.

Nowa szachownica będzie rysowana na podstawie zawartości tablicy SZAtab

Funkcję RYSUJ_SZACHY wstawiamy do pętli animacyjnej zamiast funkcji SZACHY

```
function RYSUJ_SZACHY(bok,ile){
  for (var w=1;w<=ile;w++){
  for (var k=1;k<=ile;k++){
    var x1=(w-1) * bok + 10;
    var y1=(k-1) * bok + 10;
    if (SZAtab[w][k]==0){
      KWA(x1,y1,bok,”black”,”silver”);
    }
    if (SZAtab[w][k]==1){
      KWA(x1,y1,bok,”black”,"white");
    }
  }}
};
```

POKOLENIA

Umiemy zmieniać szachownicę, a także ustawiać początkowe położenie robaczków. Aby „puścić w ruch” naszą grę należy wykonać jeszcze dwie czynności:
- wyliczyć ile robaczków znajduje się wokół każdego pola
- usunąć lub wstawić robaczki na podstawie ilości sąsiadów

Funkcja WYLICZ_SZACHY „przelatuje” po kolei przez wszystkie kratki szachownicy i w każdej z nich „rozgląda” się wokół (8 pól) i sumuje ile jest robaczków (zapalonych pól).

Zaglądamy do tablicy SZAtab
Zliczamy robaczki w zmiennej SUMA
Wynik zapisujemy do tablicy SZAsas

```
function WYLICZ_SZACHY(){
  for (var w=1;w<SZAile;w++){
  for (var k=1;k<SZAile;k++){
    var suma=0;
    suma=suma+SZAtab[w-1][k-1];
    suma=suma+SZAtab[w-1][k-0];
    suma=suma+SZAtab[w-1][k+1];
    suma=suma+SZAtab[w-0][k-1];
    suma=suma+SZAtab[w-0][k+1];
    suma=suma+SZAtab[w+1][k-1];
    suma=suma+SZAtab[w+1][k-0];
    suma=suma+SZAtab[w+1][k+1];
    SZAsas[w][k]=suma;
  }}
}
```

REGUŁY

Skoro mamy obliczoną ilość robaczków, to możemy „wyliczyć”, jakie będzie położenie nowego pokolenia robaczków.

Nowy robacze rodzi się gdy ma dokładnie trzech sąsiadów
Robaczek umiera, gdy ma mniej niż dwóch sąsiadów
Robaczek umiera, gdy ma więcej niż 3 sąsiadów

```
function WYLICZ_POKOLENIE(){
  for (var w=1;w<SZAile;w++){
  for (var k=1;k<SZAile;k++){
    if (SZAtab[w][k]==0 && SZAsas[w][k]==3) {SZAtab[w][k]=1}
    if (SZAtab[w][k]==1 && SZAsas[w][k]<=1) {SZAtab[w][k]=0}
    if (SZAtab[w][k]==1 && SZAsas[w][k]>=4) {SZAtab[w][k]=0}
  }}
}
```

START-STOP

W grze musimy mieć możliwość puszczenia machiny w ruch i jej zatrzymania w dowolnym momencie. Wstawimy do programu dodatkowy przycisk

```
<input type="button" id="BUTstart"  value="   START   " onclick=FStartStop()>
```

Funkcja, która go obsługuje ustawia zmienną ANIMUJ i zmienia napis na przycisku

Do funkcji start wstawiamy dodatkowo polecenie

```
animuj=false;
```

```
function FStartStop(){
  animuj=!animuj;
  if (!animuj) {BUTstart.value="   START   ";}
    else {BUTstart.value="   STOP   ";}
}
```

Do pętli animacyjnej polecenia

```
if (animuj){
  WYLICZ_SZACHY();
  WYLICZ_POKOLENIE();
}
```

i

GOTOWE!!!

```
<canvas id="c1" width="600" height="600"></canvas>
<br>
<input type="button" id="BUTstart"  value="   START   " onclick=FStartStop()>
<input type="range" id="SUWile" min=1 max=100 onchange=FSuwak()>
<input type="text" id="SUWtxt" size=6 onchange=FSuwak()>


<script>
var c = c1.getContext('2d')

var w = c.canvas.width;
var h = c.canvas.height;
var skok = 10;
var czas;

var MYSZKA={x:0,y:0};
var SZAWSP={x:0,y:0};

function START(){
	animuj=false;
	SZAsze=580;
	SZAile=10;
	SZAbok=SZAsze / SZAile;
	SUWile.value=SZAile;
	FSuwak();
	TABLICE();
}

function TABLICE(){
	SZAtab=new Array(101);
	SZAsas=new Array(101);
	for (var w=0; w <= 101; w++) {
		SZAtab[w] = new Array(101);
		SZAsas[w] = new Array(101);
		for (var k=0; k <= 101; k++) {
			SZAtab[w][k] = 0;
			SZAsas[w][k] = 0;
		}
	}
}

function FStartStop(){
	animuj=!animuj;
	if (!animuj) {
		BUTstart.value="   START   ";
	}
	else {BUTstart.value="   STOP   ";}
}

c.canvas.onclick =  function(evt) {
  MYSZKA.x = evt.clientX-8;
  MYSZKA.y = evt.clientY-8;
  SZAWSP.x=parseInt((MYSZKA.x - 10) / SZAbok) + 1;
  SZAWSP.y=parseInt((MYSZKA.y - 10) / SZAbok) + 1;
```

```
  SZAtab[SZAWSP.x][SZAWSP.y]=!(SZAtab[SZAWSP.x][SZAWSP.y])
}


function FSuwak(){
        SZAile=SUWile.value;
        SUWtxt.value=SZAile
        SZAbok=SZAsze / SZAile;
}

function KWA(x,y,bok,kol,wyp){
  c.lineWidth=1;
  c.fillStyle=wyp;
  c.fillRect(x,y,bok,bok);
  c.strokeStyle=kol;
  c.strokeRect(x,y,bok,bok);
}

//szachownica
function SZACHY(x,y,bok,ile,kol,wyp){
        for (var k=0; k<ile; k++){
        for (var w=0; w<ile; w++){
                KWA(x+bok*k,y+bok*w,bok,kol,wyp);
        }}
}

function RYSUJ_SZACHY(bok,ile){
  for (var w=1;w<=ile;w++){
  for (var k=1;k<=ile;k++){
    var x1=(w-1) * bok + 10;
    var y1=(k-1) * bok + 10;
    if (SZAtab[w][k]==0){KWA(x1,y1,bok,"black","silver");}
    if (SZAtab[w][k]==1){KWA(x1,y1,bok,"black","white");}
  }}
};

function WYLICZ_SZACHY(){
  for (var w=1;w<SZAile;w++){
  for (var k=1;k<SZAile;k++){
    var suma=0;
    suma=suma+SZAtab[w-1][k-1];
    suma=suma+SZAtab[w-1][k-0];
    suma=suma+SZAtab[w-1][k+1];
    suma=suma+SZAtab[w-0][k-1];
    suma=suma+SZAtab[w-0][k+1];
    suma=suma+SZAtab[w+1][k-1];
    suma=suma+SZAtab[w+1][k-0];
    suma=suma+SZAtab[w+1][k+1];
    SZAsas[w][k]=suma;
  }}
}

function WYLICZ_POKOLENIE(){
  for (var w=1;w<SZAile;w++){
  for (var k=1;k<SZAile;k++){
```

```
   if (SZAtab[w][k]==0 && SZAsas[w][k]==3) {SZAtab[w][k]=1}
   if (SZAtab[w][k]==1 && SZAsas[w][k]<=1) {SZAtab[w][k]=0}
   if (SZAtab[w][k]==1 && SZAsas[w][k]>=4) {SZAtab[w][k]=0}
  }}
}


function animacja() {
  c.clearRect(0, 0, w, h);
  c.strokeStyle="black";
  c.strokeRect(0, 0, w, h);

  RYSUJ_SZACHY(SZAbok,SZAile);
        if (animuj){
                WYLICZ_SZACHY();
                WYLICZ_POKOLENIE();
        }

  clearTimeout(czas);
  czas = setTimeout(animacja, skok);
}

START();
animacja();
</script>
```