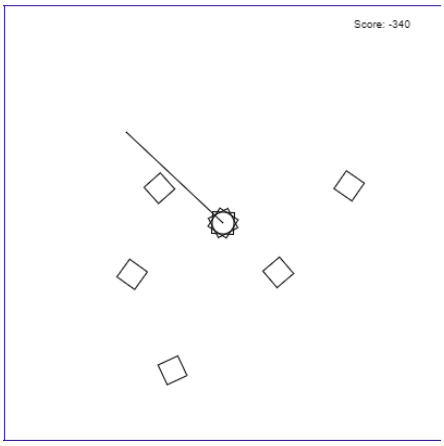
Lekcja 11 – Gra komputerowa LASER

http://www.crunchzilla.com/game-maven



Na środku ekranu kręci się laserowa baza (trzy obrócone kwadraty), z której (klikając myszką) możemy wypuszczać laserowe strzały. Z brzegów ekranu „płyną" w kierunku bazy wrogie (kwadratowe) okręty przeciwnika. Jeśli laserowa strzała trafi w kwadratowy okręt, ten zmniejsza się i znika, a my otrzymujemy punkty.

Animowane kwadraty

Program główny animacji jest typowy. Wykorzystujemy rekurencję i dwie systemowe funkcje: **clearTimeout** i **setTimeout**.

```
<canvas id="c1" width="500" height="500">
</canvas>
<script>
// definicja CANVAS i głównej animacji
  var c = c1.getContext('2d')
  var w = c.canvas.width;
  var h = c.canvas.height;
  var SKOK=50;
  var CZAS;

  function ANIMACJA(){
    c.clearRect(0,0,w,h);
    c.strokeStyle="black";
    c.strokeRect(0,0,w,h);
    // ...
    clearTimeout(CZAS);
    CZAS=setTimeout(ANIMACJA,SKOK);
  }
  ANIMACJA();
</script>
```

Obiekt KWA – kwadratowy wróg

Każdy kwadrat **KWA** posiada typowe parametry: środek (x,y), bok (bok), kąt (kat), szybkość przesuwania w pionie i poziomie (dx,dy) oraz szybkość obrotu (dk).

Metoda **przerysuj** oblicza nowe położenie i nowy kąt obrotu oraz sprawdza odbicia od brzegów. Na końcu metody wykonujemy od razu metodę **rysuj**– nie trzeba jej wykonywać podczas przerysowywania w animacji. Metoda **rysuj**, to obrócony kwadrat narysowany od środka.

```
// obiektowy kwadrat wraz z metodami
// rysuj zmiany i rysuj
function KWA(x,y,bok,kat,kol,dx,dy,dk){
  this.x=x;
  this.y=y;
  this.bok=bok;
  this.kat=kat;
  this.kol=kol;
  this.dx=dx;
  this.dy=dy;
  this.dk=dk;
  // położenie
  this.przerysuj=function(){
    var b2=this.bok/2;
    this.x=this.x+this.dx;
    this.y=this.y+this.dy;
    this.kat=this.kat+this.dk;
    if (this.x-b2 < 0 || this.x+b2 > h){
      this.dx=-this.dx;
      this.dk=-this.dk;
    }
    if (this.y-b2 < 0 || this.y+b2 > w){
      this.dy=-this.dy;
      this.dk=-this.dk;
    }
    if (this.kat>360){this.kat=0}
    //automatyczne przerysowanie
    this.rysuj();
  }
  // rysowanie
  this.rysuj=function(){
    var b2=this.bok/2;
    c.save();
    c.translate(this.x,this.y);
    c.rotate(this.kat);
    c.strokeStyle=this.kol;
    c.strokeRect(-b2,-b2,this.bok,this.bok);
    c.restore();
  }//koniec KWA rysuj
}
```

Tworzenie – losowanie

Definiujemy pustą tablicę **TKWA** i do jej kolejnych elementów wstawiamy obiekty **KWA** za pomocą instrukcji **new** i **push**.

W pętli głównej animacji wystarczy wykonać przerysowanie wszystkich obiektów w tablicy.

```
// tworzenie kwadratów i losowanie
var TKWA=[];
var maxB=30;
var maxV=4;
var maxK=4;
var ileKWA=5;

for (var i=0;i<ileKWA;i++){
  var x=Math.random()*(w-maxB)+maxB/2;
  var y=Math.random()*(h-maxB)+maxB/2;
  var kat=Math.random()*3;
  var kol="Black";
  var dx=Math.random()*maxV-1;
  var dy=Math.random()*maxV-1;
  var dk=(Math.random()*maxK-1)*0.1;
  TKWA.push(
    new KWA(x,y,maxB,kat,kol,dx,dy,dk));
}
// ...
//przerysowanie - ANIMACJA
  for (var i=0;i<TKWA.length;i++){
    TKWA[i].przerysuj();
  }
```

Inicjowanie podczas kreowania

Możemy zdefiniować metodę **init** w obiekcie **KWA**, która będzie automatycznie tworzyć nowego wroga podczas tworzenia obiektu? Co należy zmienić?

1. Funkcja KWA już nie potrzebuje parametrów
2. Nie potrzebujemy przepisywania parametrów na zmienne wewnętrzne this
3. Nowa metoda **init** będzie losować od razu do zmiennych wewnętrzny**this**
4. Gdy obiekt KWA zostanie utworzony za pomocą **New** to od razu wywołana zostanie metodę **init**
5. Przenosimy zmienne **maxB**, **maxV** i **maxK** na początek.
6. Zamiast losowania w pętli wystarczy inicjowanie obiektów KWA w pętli

```
var maxB=30;
var maxV=4;
var maxK=4;

function KWA(){
  this.init=function(){
    this.x=Math.random()*(w-maxB)+maxB/2;
    this.y=Math.random()*(h-maxB)+maxB/2;
    this.bok=maxB;
    this.kat=Math.random()*3;
    this.kol="black";
    this.dx=Math.random()*maxV-1;
    this.dy=Math.random()*maxV-1;
    this.dk=(Math.random()*maxK-1)*0.1;
  }
  this.init();
  // ...
for (var i=0; i<ileKWA; i++) {
  TKWA.push(new KWA());
}
```

Obracająca się gwiazda

Metoda **init** w funkcji-obiekcie GWIAZDA inicjuje własności gwiazdy. Po zainicjowaniu od razu jest wywoływana **this.init()**.
Metoda **przerysuj** zmienia kąt i rysuje gwiazdę **this.rysuj()**.
Metoda **rysuj** rysuje gwiazdę w punkcie (x,y), trzy obrócone o 30° kwadraty.

Po zdefiniowaniu obiektu GWIAZDA tworzymy zmienną obiektową **GWI** za pomocą instrukcji **new**.
W części głównej ANIMACJA wystarczy wywołać metodę **GWI.przerysuj()**.

```
// zmienna obiektowa
var GWI=new GWIAZDA();

// ...
// przerysowanie - ANIMACJA
GWI.przerysuj();
```

```
//obiektowa laserowa gwiazda
function GWIAZDA(){
  this.init=function(){
    this.x=w/2;
    this.y=h/2;
    this.bok=30;
    this.kat=0;
    this.kol=”blue”;
    this.dk=3;
  }
  this.init(); // inicjowanie gwiazdy
  // obliczanie i przerysowanie
  this.przerysuj=function(){
    this.kat=this.kat+this.dk;
    this.rysuj();
  }
  // rysowanie gwiazdy
  this.rysuj=function(){
    var b2=this.bok / 2;
    var b=this.bok;
    c.save();
    c.strokeStyle=this.kol;
    c.translate(this.x,this.y);
    c.rotate(Math.PI * this.kat / 180);
    c.strokeRect(-b2, -b2, b, b);
    c.rotate(Math.PI * 30 / 180);
    c.strokeRect(-b2, -b2, b, b);
    c.rotate(Math.PI * 30 / 180);
    c.strokeRect(-b2, -b2, b, b);
    c.restore();
  }
}
```

Laserowy strzał

Laserowy strzał, to linia biegnąca od środka obracającej się gwiazdy do kursora myszki. Rysowana linia jest nową metodą **strzel** w obiekcie GWIAZDA. Rysujemy wtedy, gdy przycisk myszki jest wciśnięty **MyszPrzycisk=true**, gdy puścimy przycisk myszy - linia nie jest rysowana.

Myszka – Zdarzenia

Położenie myszki zapisywane jest w zmiennej **var MYSZKA={x:0, y:0}**.

Zdarzenia związane z ruchem myszki, naciskaniem i zwalnianiem przycisku opisują zdarzenia przypisane do canvas:
onmousemove,
onmousedown,
onmouseup.

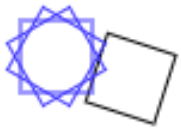
Canvas jest przesunięty względem tego, co pokazuje myszka o 8 pikseli w prawy, dolny róg.

```
// zdarzenia myszki na CANVAS
// położenie myszki
var MYSZKA = {x: 0, y: 0};
// myszka nie wciśnięta
var MyszPrzycisk = false;
// ruch myszki
c.canvas.onmousemove =  function(e) {
  MYSZKA.x = e.clientX - 8;
  MYSZKA.y = e.clientY - 8;
};
// przycisk myszki wciśnięty
c.canvas.onmousedown =  function(e) {
  MyszPrzycisk = true;
};
// przycisk myszki zwolniony
c.canvas.onmouseup = function(e) {
  MyszPrzycisk = false;
};
```

```
// laserowy strzał
// metoda w obiekcie GWIAZDA
this.strzel=function(){
  if (MyszPrzycisk){
    c.beginPath();
    c.moveTo(this.x, this.y);
    c.strokeStyle="red";
    c.lineTo(MYSZKA.x, MYSZKA.y);
    c.stroke();
  }
}

// metoda PRZERYSUJ poprawiona
this.przerysuj=function(){
  this.kat=this.kat+this.dk;
  this.rysuj();
  this.strzel();
}
```

Trafienie bazy – kwadrat uderzył w gwiazdę



Jak sprawdzić, czy gwiazda zstała trafiona przez jeden z kwadratów? Za każdym razem, gdy przerysowywany jest kwadrat trzeba będzie to sprawdzać – pętla przerysowująca kwadraty w animacji głównej. Ale jak to sprawdzić?

Obliczymy odległość pomiędzy kwadratem a bazą. Jeśli jest mniejsza niż suma połowy kwadratu i połowy bazy, to znaczy, że baza została trafiona.

```
this.CzyTrafiona=function(obiekt){
  var sx=this.x-obiekt.x;
  var sy=this.y-obiekt.y;
  var dl=Math.sqrt(sx*sx+sy*sy);
  if (dl < this.bok/2 + obiekt.bok/2){
    return true
  }
  return false;
}
```

Nową metodę **CzyTrafiona** umieścimy w obiekcie GWIAZDA. Parametrem metody będzie kwadrat **TKWA[i]**, który będzie wrzucany do zbadania w pętli przerysowującej. Jeśli gwiazda trafiona to metoda przyjmuje wartość TRUE, w przeciwnym wypadku FALSE. W pętli głównej animacji umieszczamy rysowanie punktacji.

```
for (var i=0; i<TKWA.length; i++){
  TKWA[i].przerysuj();
  if (GWI.CzyTrafiona(TKWA[i])) {
    PUNKTY=PUNKTY-10;;
  }
}
c.fillText("punkty: " + PUNKTY, w * 0.8, 20);
```

Na początku programy należy również umieścić deklarację zmiennej PUNKTY i przypisać doniej początkową wartość zero.

Alternatywne zakończenie programu – duży czerwony napis i zatrzymanie akcji – zamiast odejmowania punktów

```
c.font = '48pt sans-serif';
c.textAlign = 'center';
c.fillStyle = 'red';
c.textBaseline = 'middle';
c.fillText('GAME OVER', w / 2, h / 2);
c.restore();
exit;
```

Zestrzelenie wrogiego kwadratu

Jak sprawdzić, czy koniec laserowego strzału (położenie myszki) trafiło w środek któregoś kwadratowego wroga? Za każdym razem, gdy strzelamy obliczamy odległość pomiędzy myszką, a środkiem kwadratu. Jeśli ta odległość jest mniejsza niż połowa boku, to znaczy, że strzał się udał. Zwiększamy PUNKTY, zmniejszamy bok trafionego kwadratowego wroga, a gdy zniknie, to rodzi się nowy.

Całość w metodzie **KWA.przerysuj**.

```
if (MyszPrzycisk) {
  var dx = MYSZKA.x - this.x;
  var dy = MYSZKA.y - this.y;
  var d = Math.sqrt(dx*dx+dy*dy);
  if (d < b2 || d < 10) {
    PUNKTY=PUNKTY+1;
    this.bok=this.bok-2;
  }
}
if (this.bok <= 0) {
  this.init();  // nowy
}
```

Usuwanie wrogów z listy

Zmniejszenie wroga do zera, aby nie był widoczny, to jedno, ale dalej pozostają jego parametry w pamięci – trzeba je usunąć. Dodawanie na koniec listy realizowaliśmy za pomocą metody **push**, a usuwanie wybranego elementu metodą **splice** – główna pętla animacyjna

```
for (i=TKWA.length-1; i>=0; i=i-1) {
  if (TKWA[i].bok <= 0) {
      TKWA.splice(i, 1);
  }
}
```

Nowy wróg zawsze na brzegu

Nowy wróg powinien rodzić się losowo na brzegu. Jak to zrobić? Najpierw wylosujemy czy pojawi się na górnym-dolnym czy na lewym-prawym brzegu: **Math.random()<0.5** (połowa przypadków). Jeśli góra-dół, to Y losujemy według starych zasad, a X losujemy do momentu aż będzie na brzegu – pętla **while**. I podobnie w drugim przypadku. Polecenia umieszczamy w **KWA.init**

```
if (Math.random()<0.5) {//albo góra dół
  //Y dowolny a X losowo na brzegu
  this.y=Math.random()*(h-maxB)+maxB/2;
  do {
    var xx=Math.random()*(w-maxB)+maxB/2;
  } while (xx> maxB && xx<w-maxB);
  this.x=xx;
} else {// albo prawy lewy
  // X dowolny a Y losowo na brzegu
  this.x=Math.random()*(w-maxB)+maxB/2;
  do {
    var yy=Math.random()*(h-maxB)+maxB/2;
  } while (yy> maxB && yy<h-maxB)
  this.y=yy;
}
```

Dzieci

A gdyby tak po trafieniu wroga odrywały się od niego mniejsze wrogie kawałki? W miejscu, gdzie sprawdzamy co się dzieje po wciskaniu przycisku myszki: **KWA.przerysuj** tworzymy nowy obiekt KWA dodajemy na listę za pomocą **push**.

Okazuje się, że każdy z tych kwadracików, po zestrzeleniu rodzi nowy, duży kwadrat! Jak odróżnić zestrzelenie rodzica od zestrzelenia potomka? Każdemu dziecku nadamy dodatkowy parametr: **DZIECKO**, a gdy się rodzi nowy sprawdzamy, czy nie jest to dziecko – poprawiamy w **KWA.przerysuj**

```
if (this.bok > 10) {
  var kw = new KWA();
  kw.x = this.x + kw.dx * 5;
  kw.y = this.y + kw.dy * 5;
  kw.bok = 4;
  kw.DZIECKO = true;
  TKWA.push(kw);
}
// ...
if (this.bok <= 0 && !this.DZIECKO) {
  this.init();
}
```

Cały program

```
<canvas id="c1" width="500" height="500">
</canvas>
<script>
  var c = c1.getContext('2d')
  var w = c.canvas.width;
  var h = c.canvas.height;
  var SKOK=50;
  var CZAS;
  var maxB=30;
  var maxV=4;
  var maxK=4;
// ************************************************** KWADRAT
function KWA(){
  this.init=function(){
      // losowo na brzegach
    if (Math.random()<0.5) {
      this.y=Math.random()*(h-maxB)+maxB/2;
      do {
        var xx=Math.random()*(w-maxB)+maxB/2;
      } while (xx>maxB && xx<w-maxB);
      this.x=xx;
    } else {
      this.x=Math.random()*(w-maxB)+maxB/2;
      do {
        var yy=Math.random()*(h-maxB)+maxB/2;
      } while (yy>maxB && yy<h-maxB)
      this.y=yy;
    }

    //this.x=Math.random()*(w-maxB)+maxB/2;
    //this.y=Math.random()*(h-maxB)+maxB/2;
    this.bok=maxB;
    this.kat=Math.random()*3;
    this.kol="black";
    this.dx=Math.random()*maxV-1;
    this.dy=Math.random()*maxV-1;
    this.dk=(Math.random()*maxK-1)*0.1;
  }

  this.init();

  this.przerysuj=function(){
    var b2=this.bok/2;
    this.x=this.x+this.dx;
    this.y=this.y+this.dy;
    this.kat=this.kat+this.dk;
    // odbijanie od brzegów
      if (this.x-b2 < 0 || this.x+b2 > h){
      this.dx=-this.dx;
      this.dk=-this.dk;
    }
    if (this.y-b2 < 0 || this.y+b2 > w){
      this.dy=-this.dy;
      this.dk=-this.dk;
    }
    if (this.kat>360){this.kat=0}
      // trafienie w kwadrat
      if (MyszPrzycisk) {
      var dx = MYSZKA.x - this.x;
      var dy = MYSZKA.y - this.y;
      var d = Math.sqrt(dx*dx+dy*dy);
      if (d < b2 || d < 10) {
        PUNKTY=PUNKTY+1;
        this.bok=this.bok-2;
            // kwadrat rozpada się na mniejsze
```

```
            if (this.bok > 10) {
          var kw = new KWA();
              kw.x = this.x + kw.dx * 5;
              kw.y = this.y + kw.dy * 5;
              kw.bok = 4;
              kw.DZIECKO = true;
              TKWA.push(kw);
            }
      }
    }
      // nowy rodzi się gdy zmaleje do zera
    if (this.bok <= 0 && !this.DZIECKO) {
      this.init();
    }

    this.rysuj();
  }

  this.rysuj=function(){
    var b2=this.bok/2;
    c.save();
    c.translate(this.x,this.y);
    c.rotate(this.kat);
      c.strokeStyle=this.kol;
    c.strokeRect(-b2,-b2,this.bok,this.bok);
    c.restore();
  }
}

var ileKWA=10;
var TKWA=[];
for (var i=0; i<ileKWA; i++) {
  TKWA.push(new KWA());
}
// ****************************************************** KWADRAT
// ****************************************************** GWIAZDA
function GWIAZDA(){
  this.init=function(){
    this.x=w/2;
    this.y=h/2;
    this.bok=30;
    this.kat=0;
    this.kol="blue";
    this.dk=3;
      this.dx=0;
      this.dy=0;
  }
  //automatyczne inicjowanie własności gwiazdy
  this.init();

  this.przerysuj=function(){
      this.kat=this.kat+this.dk;
    this.rysuj();
      this.strzel();
  }

  this.rysuj=function(){
    var b2=this.bok / 2;
    var b=this.bok;
    c.save();
    c.strokeStyle=this.kol;
    c.translate(this.x,this.y);
    c.rotate(Math.PI * this.kat / 180);
    c.strokeRect(-b2, -b2, b, b);
    c.rotate(Math.PI * 30 / 180);
    c.strokeRect(-b2, -b2, b, b);
    c.rotate(Math.PI * 30 / 180);
    c.strokeRect(-b2, -b2, b, b);
      c.restore();
```

```
  }
  this.strzel=function(){
    if (MyszPrzycisk){
      c.beginPath();
      c.moveTo(this.x, this.y);
      c.strokeStyle="red";
      c.lineTo(MYSZKA.x, MYSZKA.y);
      c.stroke();
      }
  }
  // czy gwiazda trafiona przez kwadrat
  this.CzyTrafiona=function(obiekt){
    var sx=this.x-obiekt.x;
    var sy=this.y-obiekt.y;
    var dl=Math.sqrt(sx*sx+sy*sy);
    if (dl < this.bok/2 + obiekt.bok/2){
      return true
    } else { return false;}
  }
}

var GWI=new GWIAZDA();
// ************************************************** GWIAZDA
// ************************************************** MYSZKA
var MYSZKA = {x: 0, y: 0};
var MyszPrzycisk = false;

c.canvas.onmousemove =  function(e) {
  MYSZKA.x = e.clientX - 8;
  MYSZKA.y = e.clientY - 8;
};
c.canvas.onmousedown =  function(e) {
  MyszPrzycisk = true;
};
c.canvas.onmouseup = function(e) {
  MyszPrzycisk = false;
};
// ************************************************** MYSZKA
var PUNKTY=0;
// ************************************************** ANIMACJA
function ANIMACJA(){
    c.clearRect(0,0,w,h);
      c.strokeStyle="black";
    c.strokeRect(0,0,w,h);
    //rysowanie kwadratów
      for (var i=0; i<TKWA.length; i++){
      TKWA[i].przerysuj();
        //sprawdzamy vzy kwadrat trafił w gwiazdę
      if (GWI.CzyTrafiona(TKWA[i])) {
        PUNKTY=PUNKTY-10;;
      }
    }
    c.fillText("punkty: " + PUNKTY, w * 0.8, 20);
      //rysowanie gwiazdy
      GWI.przerysuj();
      //usuwanie z tabeli skasowane kwadraty
      for (i=TKWA.length-1; i>=0; i=i-1) {
      if (TKWA[i].bok <= 0) {
        TKWA.splice(i, 1);
      }
    }
    clearTimeout(CZAS);
    CZAS=setTimeout(ANIMACJA,SKOK);
}
ANIMACJA();
// ************************************************** ANIMACJA
</script>
```