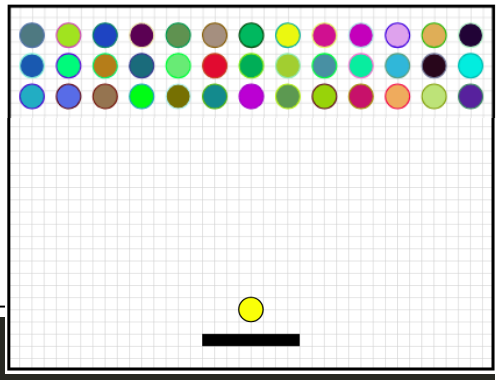
Lekcja 20 – Grafika wektorowa SVG i JS

Obiektami SVG można zarządzać również za pomocą JavaScript, co pozwala na zmianę rozmiaru, kształtu i położenia. Obliczając odległości pomiędzy nimi można także określać oddziaływania, co przydaje się do tworzenia interaktywnych prezentacji, czy gier.



Gra w kulki

Za pomocą odbijającej się kulki sterowanej klawiaturą będziemy mogli zbijać przeszkody.

```
<svg viewBox="0 0 400 300" >
  <!-- KRATKA -->
  <pattern id="kratka" patternUnits="userSpaceOnUse"
    x="0" y="0" width="10" height="10" viewBox="0 0 10 10" >
    <line x1="0" y1="0" x2="10" y2="0" stroke="lightgray" fill="none" />
    <line x1="0" y1="0" x2="0" y2="10" stroke="lightgray" fill="none" />
  </pattern>
  <!-- POLE GRY wypełnienia kraktą -->
  <rect x="0" y="0" width="100%" height="100%" fill="url(#kratka)" stroke="gray" />
  <!-- ramka pola gry -->
  <rect x="0" y="0" height="100%" width="100%" stroke="black" stroke-width="5"
fill="transparent" />
  <!-- KULA -->
  <circle id="kula" cx="200" cy="250" r="10" stroke="black" fill="yellow" />
  <!-- RAKIETKA -->
  <rect id="rakietka" x="160" y="270" height="10" width="80"/>
                <!– tutaj umieszczamy skrypt -->
</svg>
```

Atrybut **viewBox** tworzy swoisty układ współrzędnych – okienko o wymiarach 400x300 pikseli. Znacznik **pattaren** definiuje obiekt graficzny, który można przerysowywać w równych odstępach – kafelek.

Skrypty

Pod instrukcjami rysującymi elementy ekranu umieszczamy szablon skryptu gry, w którym znajdują się szkielety funkcji. Zmienne **Vrakietka**, **Vkula** oraz **Vmur** zawierają uchwyty do obiektów odpowiednich SVG. Zauważ, że **Vmur** wskazuje na kontener **g** o **id=”mur”**, który nie jest jeszcze gotowy. Pola **vh** i **vv** obiektu **kula** opisują szybkość poruszania się kuli po planszy. Tablica **przeszkody** będzie przechowywać elementy przeszkód dla kuli, które znajdują się w kontenerze **mur**. Zauważ także, że po załadowaniu strony, wywoływana jest funkcja **setInterval**, odpowiadająca za cykliczne wywołanie funkcji gry **ANIMACJA**.

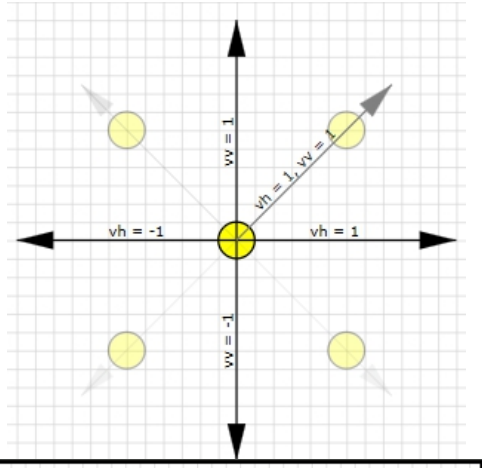
```
<g id="mur" />
<script type="text/ecmascript">
  var Vrakietka = document.getElementById("rakietka");
  var Vkula = document.getElementById("kula");
  var Vmur = document.getElementById("mur");
  var przeszkody = [];
  Vkula.vh = 2;
  Vkula.vv = -2;
  function Losuj(min, max) {
    return Math.floor(Math.random()*(max-min+1))+min;
}
function ANIMACJA() {
  SprawdzajKolizje();
  KolizjaRakietka();
  KolizjaMur();
}
function KolizjaRakietka() { }
function KolizjaMur() { }
function BudujMur() { }
function SprawdzajKolizje() { }

BudujMur();
setInterval('ANIMACJA()',16.7);
</script>
```

Uwaga. Co prawda wszyscy posługują się poleceniem **dokument.getElementById()** do tworzenia zmiennych obiektowych, ale równie dobrze można zastosować uproszczoną wersję: **var Vmur=”mur”;**

Piłka w ruchu

Dla uproszczenia piłka będzie poruszać się w czterech skośnych kierunkach określonych przez pola **vh** i **vv**. Położenie środka kuli odczytujemy z pomocą: **kula.cx.baseVal.value** i **kula.cy.baseVal.value**. Po dodaniu dwóch wierszy kodu do funkcji ANIMACJA nasza kulka zacznie się poruszać.




```
Vkula.cx.baseVal.value += Vkula.vh;
Vkula.cy.baseVal.value += Vkula.vv;
```

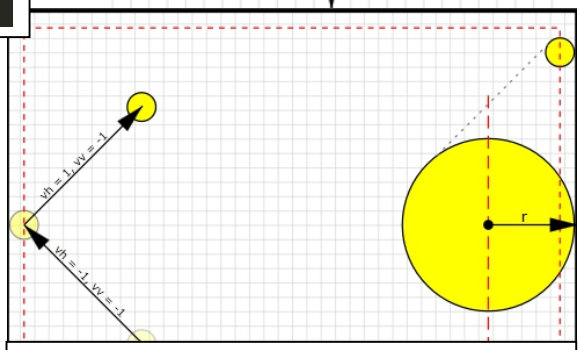
W tym miejscu ujawnia się siła składni języka C. Zapisanie tych linijek w tradycyjny sposób byłoby bardziej pracochłonne.

```
var kx=Vkula.cx.baseVal.value;
var ky=Vkula.cy.baseVal.value;
kx = kx + Vkula.vh;
ky = ky + Vkula.vv;
Vkula.cx.baseVal.value = kx;
Vkula.cy.baseVal.value = ky;
```

Kolizje z brzegami

Sprawdzamy czy środek kulki powiększony o promień styka się z brzegiem pola gry. Ten sam efekt da pomniejszenie pola gry o promień kuli. W instrukcjach warunkowych sprawdzamy jednocześnie odbijanie od dwóch przeciwległych brzegów – gdy kula osiąga brzeg, to szybkość kuli w pionie lub poziomie zmienia się na przeciwną.

Wiersze kodu wklejamy do funkcji **SprawdzajKolizje**. W zmiennej **r** znajduje się promień kuli.




```
var r=Vkula.r.baseVal.value;
var cx=Vkula.cx.baseVal.value;
var cy=Vkula.cy.baseVal.value;
if ((cx+r >= 400) || (cx-r <= 0)) {
  Vkula.vh = -Vkula.vh;
}
if ((cy+r >= 300) || (cy-r <= 0)) {
  Vkula.vv = -Vkula.vv;
}
```

Sterowanie rakietką

Funkcja **processKey** za pomocą pola **keyCode** zwraca numer naciśniętego klawisza: klawisz kursora w lewo – kod 39, klawisz kursora w prawo – kod 37.

Metoda **window.addEventListener** dodaje funkcję do obsługi zdarzeń przeglądarki. Funkcję dodajemy do szablonu skryptu w dowolnym miejscu.

```
function processKeys(e) {
  switch (e.keyCode) {
   case 39 : // <-
     var x = Vrakietka.x.baseVal.value;
     x = x + 20;
     if (x > 310) x = 310;
     Vrakietka.x.baseVal.value = x;
     break;
   case 37 : // ->
     var x = Vrakietka.x.baseVal.value;
     x = x - 20;
     if (x < 10) x = 10;
     Vrakietka.x.baseVal.value = x;
     break;
}}
window.addEventListener('keydown', processKeys, false);
```

Odbijanie od rakietki

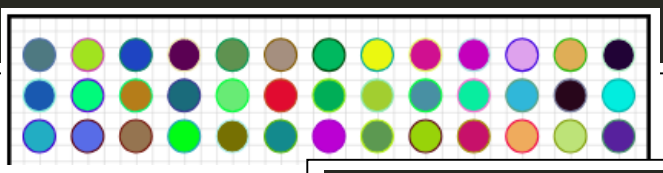
Pobieramy położenie rakietki (pX, pY), szerokość rakietki (pW) oraz położenie kulki (cX, cY) i jej promień cR z obiektów SVG. Sprawdzamy kolizję kulki z rakietką w podobny sposób jak podczas kolizji z murem – jeśli kula powiększona o promień styka się z rakietką, to zmieniamy kierunek poruszania się kuli, ale tylko w pionie.

Wiersze kodu wstawiamy do funkcji **KolizjaRakietka.**

```
var pX = Vrakietka.x.baseVal.value;
var pY = Vrakietka.y.baseVal.value;
var pW = Vrakietka.width.baseVal.value;
var cX = Vkula.cx.baseVal.value;
var cY = Vkula.cy.baseVal.value;
var cR = Vkula.r.baseVal.value;
if (cY+cR>pY && cY<pY) {
if (cX>pX && cX<pX+pW) {
  Vkula.vv = -Vkula.vv;
}}
```

Budowanie muru
Trzy rzędy po 13 kolorowych kulek rozmieszczamy równomiernie na planszy. Kolory są ustawiane losowo.
createElementNS - tworzy nowy element SVG z bazy gotowych elementów
przeszkody.push(prz) – dodaje nowy element do tablicy
mur.appendChild(prz) – dodanie nowego elementu do graficznego elementu muru na stronie.
Fragment kodu wstawiamy do funkcji **BudujMur**.

```
for (var i = 0; i < 13; i++) {
for (var j = 0; j < 3; j++) {
  var prz = document.createElementNS("http://www.w3.org/2000/svg", "circle");
  prz.r.baseVal.value = 10;
  prz.cx.baseVal.value = i*30+20;
  prz.cy.baseVal.value = j*25+25;
  prz.style.fill = 'rgb(' + Losuj(0,255) + ',' + Losuj(0,255) + ',' + Losuj(0,255)+')';
  prz.style.stroke = 'rgb(' + Losuj(0,255) + ',' + Losuj(0,255) + ',' + Losuj(0,255)+')';
  przeszkody.push(prz);
  Vmur.appendChild(prz);
}}
```



Koniec gry
Piłka zatrzymuje się gdy dotknie dolnej krawędzi muru. Zmienić musimy warunki odbicia tylko od dolnej krawędzi w funkcji **KolizjaMur**.

```
if (cy+r >= 300) {
  Vkula.vh = 0;
  Vkula.vv = 0;
}
if (cy-r <= 0) {
  Vkula.vv = -Vkula.vv;
}
```

Kolizje z kolorym murem
Przeglądamy w pętli całą tablicę **przeszkody**. Do zmiennej **prz** wstawiamy kolejny element tablicy **przeszkody**. Zmienne **deltaX** i **deltaY** służą do obliczenia odległości pomiędzy kulą a kolorowym elementem muru – wykorzystujemy twierdzenie Pitagorasa.

```
var isEmpty = true;
for (var i = przeszkody.length - 1; i >= 0; i--) {
  var prz = przeszkody[i];
  if (prz == null) continue;
  isEmpty = false;
  var deltaX = prz.cx.baseVal.value - Vkula.cx.baseVal.value;
  var deltaY = prz.cy.baseVal.value - Vkula.cy.baseVal.value;
  var d = Math.sqrt( (deltaX*deltaX) + (deltaY*deltaY) );
  if (d <= ( prz.r.baseVal.value + Vkula.r.baseVal.value )) {
    Vmur.removeChild(prz);
    przeszkody[i] = null;
    if (deltaX >= 0) Vkula.vh=-Vkula.vh;
    if (deltaY >= 0) Vkula.vv=-Vkula.vv;
        return;
}}
if (isEmpty) BudujMur();
```

Jeśli obliczona odległość **d** jest mniejsza niż suma średnic kuli i kolorowej kulki do usuwamy element muru z graficznego obiektu – **mur.removeChild(prz)** oraz z tablicy – **przeszkody[i]=null**. Odbijamy również kulkę – zmieniając kierunek jej ruchu. Jeśli zbito wszystkie kulki budujemy mur od nowa.

UWAGA Jeśli zapisujemy dokument z rozszerzeniem HTML, to skrypt w połączeniu z JS jakoś sobie radzi. Gdy chcemy zapisać z rozszerzeniem SVG (i wstawić na przykład jako obraz), to skrypt uzupełniamy o wiersze zaznaczone zielonym kolorem.

```
<svg viewBox="0 0 400 300"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
...
<script>
<![CDATA[
...
]]>
</script>
</svg>
```