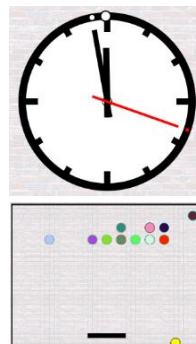


## JS 02 – Grafika wektorowa SVG na stronach WWW (13)

Grafika tworzona za pomocą składni SVG jest grafiką wektorową – oznacza to, że utworzone w ten sposób obiekty można powiększać bez utraty jakości obrazu. Kiedy stosować CANVAS, a kiedy SVG? SVG, to matematyczny opis tego co znajdzie się na stronie. Canvas, to obraz składający się z pojedynczych pikseli. SVG nie powinien być wykorzystywany, gdy obraz jest skomplikowany i dynamiczny (np. gry) – procesor jest zbyt zajęty analizą obrazu. SVG nadaje się do tworzenia elementów, które wymagają interakcji z użytkownikiem, np. przycisków zmieniających swój wygląd w zależności od położenia myszki oraz do obrazów wymagających skalowania bez utraty jakości (np. wykresy). Obie techniki nadają się do tworzenia animacji za pomocą JavaScript, choć w przypadku Canvas należy przerysować cały obrazek, a w SVG jedynie zmieniać element.



**Pamiętaj o tym, by zrzut ekranu DOKUMENTOWAŁ Twoją pracę**

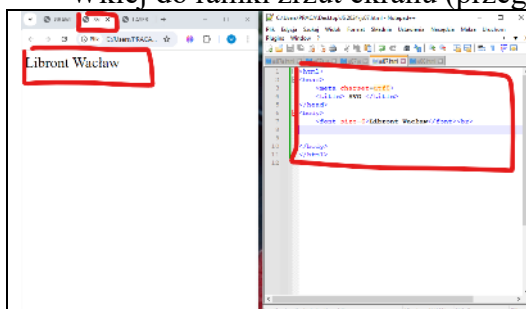
### Plik (1)

- W swoim folderze utwórz nowy dokument: **js07.html**
- Otwórz dokument w notatniku i w przeglądarce
- Do dokumentu **HTML** wklej tekst z ramki


```
<html>
<head>
  <meta charset=utf8>
  <title> SVG </title>
</head>
<body>
  <font size=6>Libront Waclaw</font><br>
</body>
</html>
```

Szablon strony WWW

- Zmień tytuł witryny - **inicjały** i wpisz swoje **nazwisko i imię**
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



### Kwadraty i koła (1)

- Do dokumentu **HTML**, przed znacznik **</body>**  wklej tekst z ramki

```
<svg width=200 height=200>
  <rect x=0 y=0 width=95 height=95 fill=red stroke=black />
  <rect x=105 y=105 width=95 height=95 fill=green stroke=black />
  <rect x=105 y=0 width=95 height=95 fill=blue stroke=black />
  <rect x=0 y=105 width=95 height=95 fill=yellow stroke=black />
  <circle cx=100 cy=100 r=60 fill=white stroke=black />
</svg>
```

znaczniki SVG możemy wklejać w dowolne miejsce kodu HTML

obszar SVG określamy za pomocą parametrów width i height

rect – prostokąty – parametry są współrzędne lewego górnego rogu, szerokość, wysokość, kolor wypełnienia i ramki

circle – koła – parametry są współrzędne środka, promień, kolor wypełnienia i ramki

#### ZADANIE

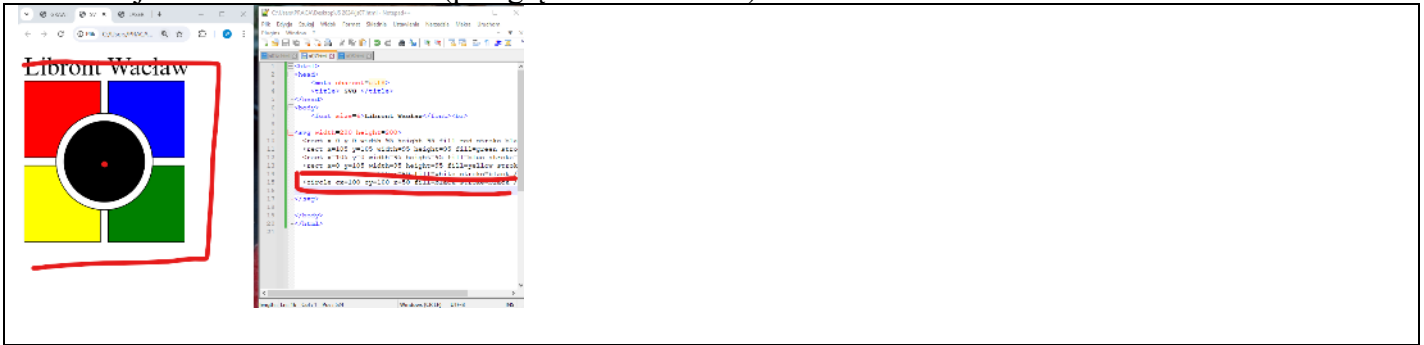
Instrukcja `<circle cx=100 cy=100 r=60 fill=white stroke=black />`

rysuje białe koło z czarnym brzegiem, o środku w punkcie (100,100) i promieniu 60

- Na środku białego koła narysuj **czarne koło** o promieniu **50**

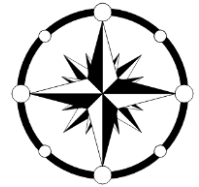
instrukcję wstaw przed znacznik `</svg>`

- Zapisz dokumenty i odśwież przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Róża wiatrów - Obracanie (1)

Obiekty svg można transformować, tzn. przesuwać, skalować i obracać. Czasem łatwiej narysować standardowy obiekt, a potem go przekształcić, np., obrócić niż zastanawiać się, jakie współrzędne wymyślić, aby obiekt był pochylony. Tworząc symbol róży wiatrów, w praktyczny sposób zastosujemy te trzy przekształcenia.



- Ustaw kolor strony zmieniając znacznik `<body>` `<body bgcolor=silver>`
- Do dokumentu HTML, przed znacznik `</body>` wklej tekst z ramki

```

<br>
<svg width=300 height=300>
<g id=ramie stroke=black>
  <polygon points="150 30, 150 150 125,125" fill=black />
  <polygon points="150 30, 150 150 175,125" fill=white />
  <circle cx=150 cy=16 r=10 fill=white stroke=black />
</g>
</svg>

```

kwadratowy obszar SVG o boku 300  
 znacznik `<g>` definiuje grupę, która składa się z dwóch wielokątów  
 grupa ma opisany identyfikator, który będzie niezbędny podczas transformacji  
 wszystkie obiekty grupy mają czarny kolor ramki  
 polygon – wielokąt składający się z odcinków, współrzędne x i y podajemy po przecinku

- (1) Na środku białego koła narysuj **czarne kółko** o promieniu 6  
`<circle cx=150 cy=16 r=6 fill=black stroke=black />`

instrukcję wstaw do grupy `</g>`

- Przed znacznik `</svg>` wklej tekst z ramki

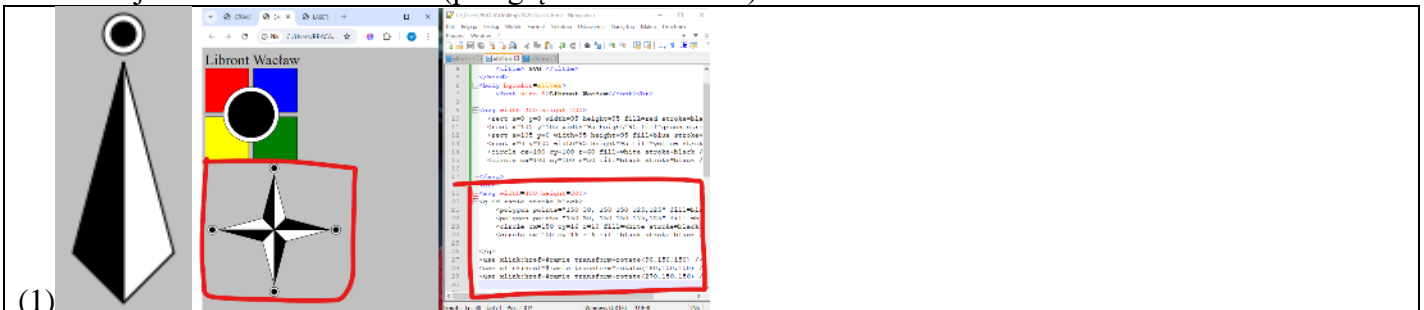
```

<use xlink:href=#ramie transform=rotate(90,150,150) />
<use xlink:href=#ramie transform=rotate(180,150,150) />
<use xlink:href=#ramie transform=rotate(270,150,150) />

```

`use xlink:href=#ramie` używamy grupy ramie  
`transform=rotate()` transformacja poprzez obrót  
`(90,150,150)` parametrami rotacji są kąt i środek obrotu

- Zapisz dokumenty i odśwież przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Róża wiatrów - Skalowanie (1)

Kolejność instrukcji ma znaczenie – kolejne polecenia rysują na wierzchu przysłaniając poprzednie rysunki  
Skalujemy względem lewego, górnego rogu obiektu. Aby przeskalowany rysunek znajdował się w środku, należy go dodatkowo przesunąć. O ile? Możesz zastosować równanie:

$przesuniecie(x,y) = (nowy(x,y) - stary(x,y) * skala) / skala$

STARY środek obrotu znajduje się w punkcie (150,150) i po przesunięciu na NOWY punkt też ma znajdować się w punkcie (150,150). Pomniejszamy gwiazdę w skali 0,75. Nasze równanie przyjmie postać:  $przesuniecie(x,y) = ((150,150) - (150,150)*0.75)/0.75 = (50,50)$  – więc przesuwamy o 50 pikseli.

```
<svg width=300 height=300>
```

- Wpisz instrukcję z ramki `<g id=ramie stroke=black>`

```
<circle cx=150 cy=150 r=135 stroke-width=12 stroke=black fill=white />
```

jako pierwsza instrukcja svg, aby nie przysłoniło róży wiatrów

stroke-width – szerokość ramki

```
<use xlink:href=#ramie transform=rotate(90,150,150) />
```

```
<use xlink:href=#ramie transform=rotate(180,150,150) />
```

- Usuń trzy wiersze `<use xlink:href=#ramie transform=rotate(270,150,150) />`

i wklej instrukcje z ramki

```
<g id=gwiazda>
<use xlink:href=#ramie />
<use xlink:href=#ramie transform=rotate(90,150,150) />
<use xlink:href=#ramie transform=rotate(180,150,150) />
<use xlink:href=#ramie transform=rotate(270,150,150) />
</g>
```

utworzono grupę, która rysuje cztery ramiona

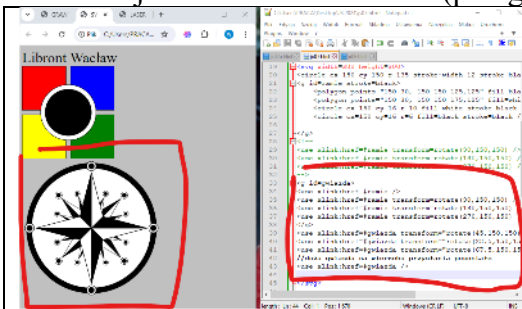
- Wklej tekst z ramki przed znacznik `</svg>`

```
<use xlink:href=#gwiazda transform="rotate(45,150,150) scale(0.75) translate(50,50)" />
<use xlink:href=#gwiazda transform="rotate(22.5,150,150) scale(0.6) translate(100,100)" />
<use xlink:href=#gwiazda transform="rotate(67.5,150,150) scale(0.6) translate(100,100)" />
//duża gwiazda na wierzchu przysłania pozostałe
<use xlink:href=#gwiazda />
```

skalujemy gwiazdę 3 razy i na końcu rysujemy dużą gwiazdę

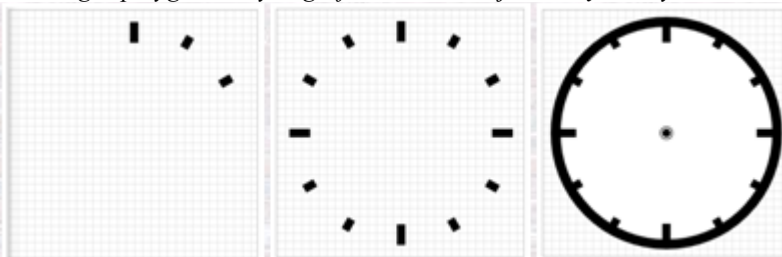
przesunięcia w czasie skalowania obliczone za pomocą wzoru:  $x=(x2-x1)*skala/skala$

- Zapisz dokumenty i odśwież kilka razy przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Zegar - Tarcza (1)

Zegar przygotowany w grafice wektorowej SVG będzie wyświetlał aktualny czas.



obszar svg ma wymiary 300 na 300 (środek w punkcie (150,150))

kreski godzin (duża i dwie mniejsze) rysujemy w jednym miejscu (145,15), a potem dwie mniejsze obracamy o 30 i 60 stopni  
taką grupę trzech kresek obracamy trzy razy o 90, 180 i 270 stopni

`</svg>`

`</body>`

- Do dokumentu **HTML**, przed znacznik `</body>` `</body>` wklej tekst z ramki

```
<svg width=300 height=300>
  <!-- tarcza zegara - koło -->
  <circle cx=150 cy=150 r=135 stroke-width=12 stroke=black fill=white />
  <!-- 3 punkty godzin: większy, 5 i 10 - mniejsze -->
  <g id=trzy >
    <rect x=145 y=15 width=10 height=25 />
    <rect x=145 y=15 width=10 height=15 transform=rotate(30,150,150) />
    <rect x=145 y=15 width=10 height=15 transform=rotate(60,150,150) />
  </g>

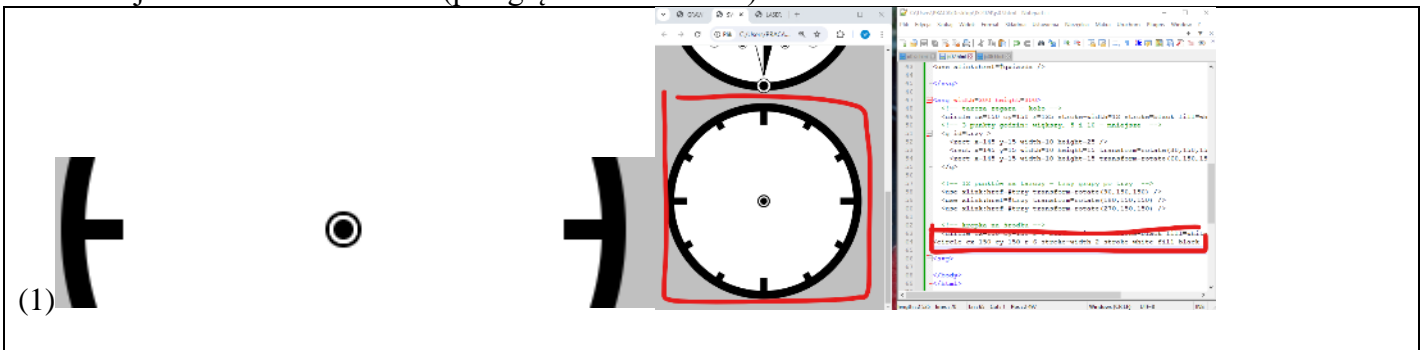
  <!-- 12 punktów na tarczy - trzy grupy po trzy -->
  <use xlink:href=#trzy transform=rotate(90,150,150) />
  <use xlink:href=#trzy transform=rotate(180,150,150) />
  <use xlink:href=#trzy transform=rotate(270,150,150) />

  <!-- kropka na środku -->
  <circle cx=150 cy=150 r=8 stroke-width=2 stroke=black fill=white />
</svg>
```

najpierw narysowane duże koło tarczy, żeby nie przysłoniło pozostałych elementów

`<circle cx=150`

- (1) Do białego kółka na środku tarczy `<svg>` wstaw czarne koło o promieniu 6
- Zapisz dokumenty i odśwież przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)

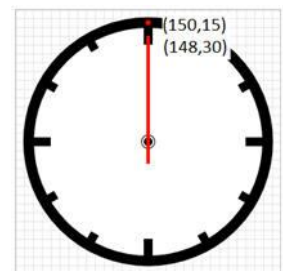


## Zegar - Wskazówki (1)

Wskazówka sekundowa jest prostokątem i przechodzi przez środek tarczy (150,150). Ma szerokość 4, więc powinna być przesunięta o 2 piksele w lewo, do punktu (148).

Do każdej wskazówki dorysujemy małe kółeczko na końcu, które będzie przesuwalo się wraz z nią na brzegu tarczy. Wskazówki grupujemy i nadajemy im identyfikatory: WSKsek, WSKmin, WSKgod, które będą niezbędne podczas animacji.

Wskazówek powinny być rysowane na tarczy zgodnie z kolejnością, ale przed kropkami na środku.



`<circle cx=150`

`<svg>`

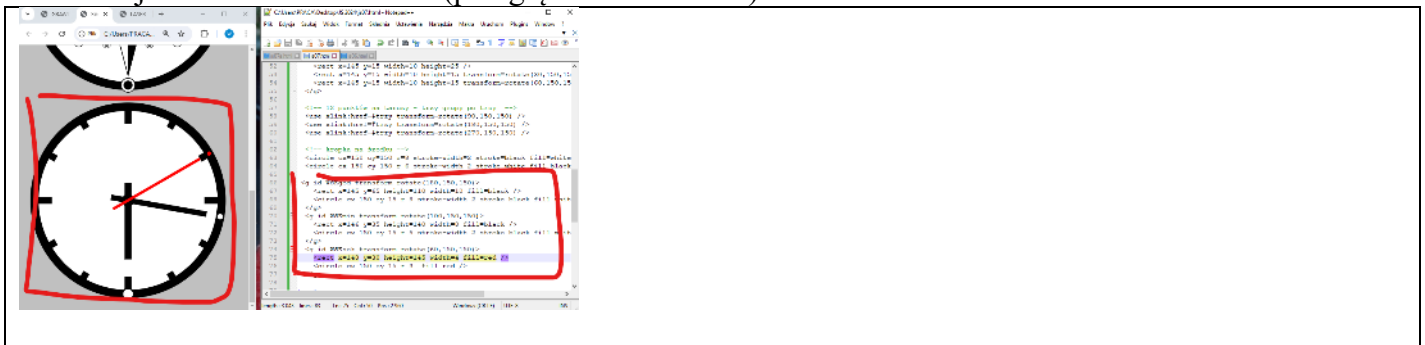
- Do dokumentu **HTML**, przed znacznik `</svg>` w zegarze `<svg>` wklej instrukcje

```
<g id=WSKgod>
  <rect x=145 y=65 height=110 width=10 fill=black />
  <circle cx=150 cy=15 r=8 stroke-width=2 stroke=black fill=white />
</g>
<g id=WSKmin>
  <rect x=146 y=35 height=140 width=8 fill=black />
  <circle cx=150 cy=15 r=5 stroke-width=2 stroke=black fill=white />
</g>
<g id=WSKsek>
  <rect x=148 y=30 height=145 width=4 fill=red />
  <circle cx=150 cy=15 r=3 fill=red />
</g>
```

najpierw wskazówka godzinowa - grubość 10 długość 110 – na spodzie  
w środku wskazówka minutowa - grubość 8 długość 140  
na wierzchu wskazówka sekundowa - grubość 4 długość 145 kolor czerwony

- Zapisz dokument i odśwież przeglądarkę

- Obróć wskazówkę godzinową o  $180^\circ$  dopisując polecenie transformacji  
`<g id=WSKgod transform=rotate(180,150,150)>`  
 wokół punktu (150,150) obracamy obiekt o  $180^\circ$
- Wskazówkę minutową obróć w ten sam sposób o  $100^\circ$  a sekundową o  $60^\circ$   
 Zapisz dokument i odśwież przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Zegar - Animacja (1)

Wykorzystamy JavaScript i opisywaną już metodę rekurencyjnego wykonywania instrukcji: `setInterval()`

- Do dokumentu **HTML**, przed znacznik `</body>` wpisz skrypt animacyjnyjny

```
<script>
  var skok = 1000;
  var czas;
  function ANIMACJA() {
    clearTimeout(czas);
    czas=setInterval(ANIMACJA,skok);
  }
  ANIMACJA();
</script>
```

funkcja ANIMACJA będzie wykonywała się co 1000 milisekund (1 sekundę)

```
var czas;
function ANIMACJA() {
```

- Do dokumentu **HTML**, przed funkcją ANIMACJA, wklej tekst z ramki

```
var Wsek=document.getElementById("WSKsek");
var Wmin=document.getElementById("WSKmin");
var Wgod=document.getElementById("WSKgod");
```

do zmiennych Wsek,Wmin,Wgod parametry obiektów SVG za pomocą znanego polecenia: `document.getElementById(ID)`

```
function ANIMACJA() {
```

- Do dokumentu **HTML** do wnętrza funkcji ANIMACJA, wklej tekst z ramki

```
var date = new Date();
var Dsek = date.getSeconds();
var Dmin = date.getMinutes();
var Dgod = date.getHours() % 12;
Wsek.setAttribute('transform','rotate('+(Dsek * 6)+'',150,150)');
Wmin.setAttribute('transform','rotate('+(Dmin * 6)+'',150,150)');
Wgod.setAttribute('transform','rotate('+(Dgod * 30)+'',150,150)');
```

w zmiennej date mamy aktualny czas

w zmiennych Dsek, Dmin i Dgod pobieramy ze zmiennej date sekundy, minuty i godziny

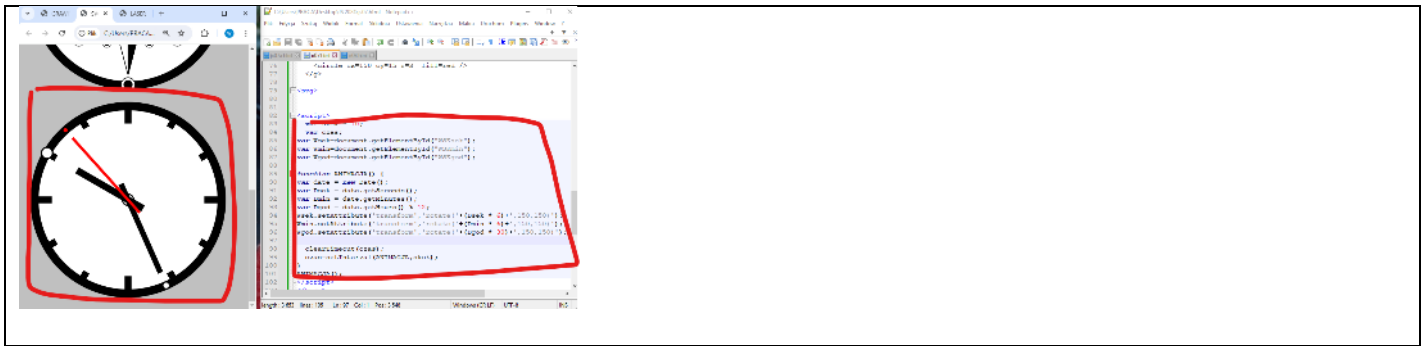
godziny modulo 12,, bo na zegarze jest 12 godzin

ustawiamy atrybuty obiektów svg

wskazówki godzinowa obraca się co 30 stopni

wskazówki minutowa i sekundowa obracają się co 6 stopni

- Zapisz dokumenty i odśwież przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Zegar - Wskazówki (1)

- Zmień skok animacji `var skok = 10;`
- Wstaw do funkcji ANIMACJA() `Wsek.setAttribute('transform', 'rotate('+(D` zmienną związaną z milisekundami `var Dmil = date.getMilliseconds();`
- Usuń z funkcji ANIMACJA() instrukcje rysujące wskazówki  
`Wsek.setAttribute('transform', 'rotate('+(Dsek * 6)+'', 150, 150)');`  
`Wmin.setAttribute('transform', 'rotate('+(Dmin * 6)+'', 150, 150)');`  
`Wgod.setAttribute('transform', 'rotate('+(Dgod * 30)+'', 150, 150)');`

i wklej nowe z ramki

```
Wsek.setAttribute('transform', 'rotate('+(Dsek * 6 + Dmil * 6/1000)+'', 150, 150)');
Wmin.setAttribute('transform', 'rotate('+(Dmin * 6 + Dsek * 6/60)+'', 150, 150)');
Wgod.setAttribute('transform', 'rotate('+(Dgod * 30 + Dmin * 6/12)+'', 150, 150)');
```

wykonywanie funkcji przestawione na 10 milisekund

Dmin - dodatkowa zmienna, która pobiera ze zmiennej data milisekundy

sekundowa obraca się co 10 milisekund

minutowa obraca się co 1 sekundę

godzinowa obraca się co 1 minutę

- Zmień kolor tarczy na niebieski  
`stroke=blue`
- Zmień punkty na tarczy na niebieski  
trzy razy wpisz `fill=blue`
- Zapisz dokumenty i przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Bubble - Pole gry (1)

- Do dokumentu HTML, za znacznikiem `<font size=6>Libront` wklej tekst z ramki

```
<svg width=400 height=300>
  <rect x=0 y=0 width=400 height=300 fill=grey />
  <rect x=0 y=0 height=300 width=400 stroke=black stroke-width=5 fill=transparent />
  <circle id=kula cx=200 cy=250 r=10 stroke=black fill=black />
  <rect id=rakietka x=160 y=270 height=10 width=80 />
</svg>
<br><br>
```

pole gry ma wymiary 400x300 w kolorze szarym

ramka jest czarnym prostokątem wypełnienie przezroczyste

żółta kulka i czarna prostokątna rakieta z własnym ID, które posłużą do komunikacji z JS

- Zmień kolor kulki na żółty `fill=yellow`
- Zmień kolor rakiety na niebieski `fill=blue`
- Zapisz dokumenty i odśwież przeglądarkę
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Bubble - Odbicia (1)

- Do dokumentu **HTML**, przed znacznik `</svg>` wpisz tekst `<g id=mur > </g>`  
grupa mur będzie dynamicznie – za pomocą JS wypełniana kulkami, ale musi być zadeklarowana wcześniej

```
<rect id=rakieta x=160 y=270  
</svg>  
<br><br>
```

- Do dokumentu **HTML**, za znacznikiem `</svg>` wklej tekst z ramki

```
<script>  
var Vrak = document.getElementById("rakieta");  
var Vkul = document.getElementById("kula");  
var Vmur = document.getElementById("mur");  
var przeszkody = [];  
Vkul.vh = 2;  
Vkul.vv = -2;  
  
function Losuj(min, max) {  
    return Math.floor(Math.random() * (max-min+1)) + min;  
}  
  
function BudujMur() { }  
function SprawdzajKolizje() { }  
function processKeys(e) { }  
  
function BUBLE() {  
    SprawdzajKolizje();  
}  
  
BudujMur();  
setInterval(BUBLE, 10);  
</script>
```

szkielet programu

zmiennne `Vrak`, `Vkul`, `Vmur` zawierają uchwyty do obiektów w SVG  
pola `vh` i `vv` obiektu `Vkul` opisują szybkość poruszania się kuli po planszy  
tablica `przeszkody` będzie przechowywać kulki, które znajdują się w grupie mur  
sposób animowania opisany wcześniej – rekurencyjne wykonywanie funkcji `BUBLE`

```
function BUBLE() {  
    SprawdzajKolizje();  
}
```

- Do dokumentu **HTML**, funkcja `BUBLE`, wklej tekst z ramki

```
var kx=Vkul.cx.baseVal.value;  
var ky=Vkul.cy.baseVal.value;  
kx = kx + Vkul.vh;  
ky = ky + Vkul.vv;  
Vkul.cx.baseVal.value = kx;  
Vkul.cy.baseVal.value = ky;  
var kr=Vkul.r.baseVal.value;
```

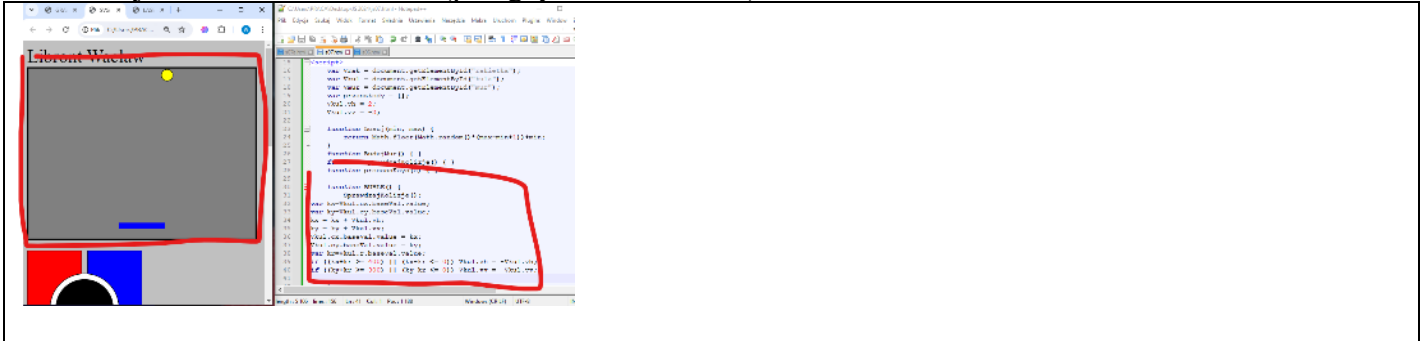
```

if ((kx+kr >= 400) || (kx-kr <= 0)) Vkul.vh = -Vkul.vh;
if ((ky+kr >= 300) || (ky-kr <= 0)) Vkul.vv = -Vkul.vv;

```

do  $kx$  i  $ky$  pobieramy aktualną pozycję środka żółtej kuli  
 zwiększamy  $kx$  i  $ky$  o prędkości  $vx$  i  $vy$   
 po obliczeniu nowych pozycji, zapisujemy je w obiekcie  $Vkul$   
 zmienna  $kr$  przechowuje promień kuli  
 jeżeli kulka dotrze do brzegu (środek + promień) to zmieniamy kierunek ruchu

- Zapisz dokumenty i odśwież przeglądarkę, **gdy kulka znajduje się w górnej części pola gry**
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



## Bubble - Rakiетка (1)

Funkcja `processKey` za pomocą pola `keyCode` zwraca numer naciśniętego klawisza (np. klawisz kursora w lewo, to 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. Jeśli rakiетка nie chce się poruszać - kliknij myszką w pole gry (musi być zaznaczone).

- W dokumencie **HTML**, usuń funkcję `function processKeys(e) { }` i zastąp ją poleceniami z ramki

```

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

```

`e.keyCode` sprawdzamy, co wciśnięto na klawiaturze  
 kod 39 to strzałka w lewo, kod 37 to strzałka w prawo  
 do zmiennej  $x$  pobieramy z obiektu  $SVG$   $Vrak$  pozycję rakiетки  
 zwiększamy lub zmniejszamy o 20 i sprawdzamy, czy nie wychodzi poza ramkę

```
if ((ky+kr >= 300) || (ky-kr <= 0))
```

- Dokument **HTML**, funkcja **BUBLE**,  
 wpisz instrukcje z ramki  
 odbijanie kulki od rakiетки

```

var rx = Vrak.x.baseVal.value;
var ry = Vrak.y.baseVal.value;
var rw = Vrak.width.baseVal.value;
if (ky + kr > ry && ky < ry)
  if (kx > rx && kx < rx + rw)
    Vkul.vv = -Vkul.vv;

```

zmiennne  $rx$  i  $ry$  pobierają położenie rakiетки  
 zmienna  $rw$  pobiera szerokość rakiетки  
 jeżeli środek kulki powiększony o promień styka się z rakietką (na wysokości i szerokości)  
 to zmieniamy szybkość kulki na przeciwną



- Zapisz dokumenty i odśwież przeglądarkę  
*kulka powinna odbijać się od rakiетки*
- **Przesuń rakiетки do prawego brzegu pola gry i poczekaj aż kulka odbije się od rakiетки**
- **Wklej do ramki zrzut ekranu (przeglądarka i notatnik)**



## Bubble - Mur (1)

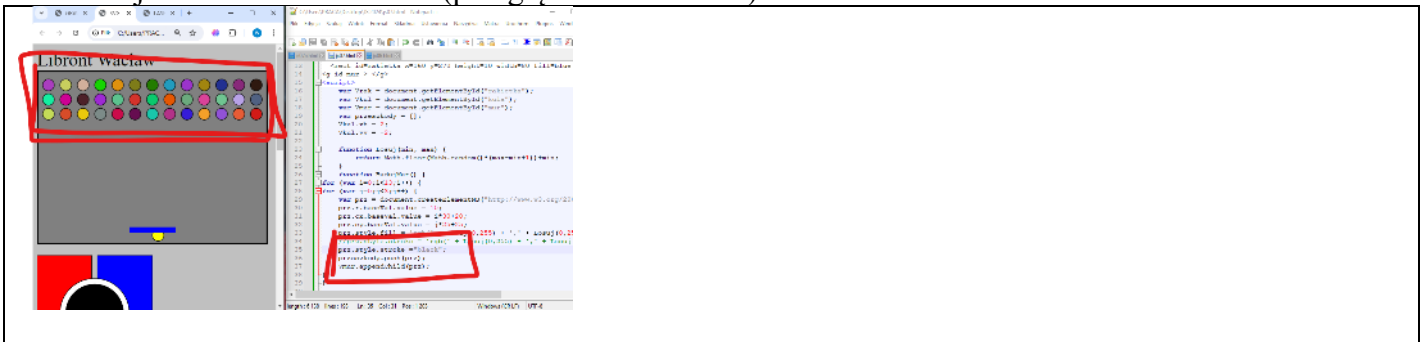
3 rzędy po 13 kolorowych kulek rozmieszczamy równomiernie na planszy. Kolory są ustawiane losowo.

- W dokumencie **HTML**, zastąp funkcję `function BudujMur() { }` tekstem z ramki

```
function 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);
}}
}
```

dwie pętle indeksują trzy rzędy po 13 kul  
createElementNS tworzy nowy element SVG z bazy gotowych elementów  
zmienna prz jest obiektem, którego wygląd pobieramy z internetu  
obiektowi prz przypisujemy promień pozycję kolor wypełnienia i ramki  
kolory losujemy za pomocą funkcji i sklejamy w zmienną rgb()  
przeszkody.push(prz); - tablica przeszkody powiększa się o jeden element  
Vmur.appendChild(prz); - narysowanie elementu muru na stronie – dodanie do obiektu Vmur elementu prz

- Kulki mają kolorowe brzegi
- **Ustaw brzegi wszystkich kulek na czarny kolor wpisując w odpowiednie miejsce polecenie**  
`prz.style.stroke = "black";`
- Zapisz dokumenty i odśwież przeglądarkę
- **Wklej do ramki zrzut ekranu (przeglądarka i notatnik)**



## Bubble - Kolizja z murem (1)

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. 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. Kod wstawiamy do funkcji SprawdzajKolizje.

- W dokumencie **HTML**, zastąp funkcję `function SprawdzajKolizje() { }` tekstem z ramki

```
function SprawdzajKolizje() {
var koniec = true;
for (var i=przeszkody.length-1;i>=0;i--) {
    var prz = przeszkody[i];
    if (prz == null) continue;
    koniec = false;
    var dX = prz.cx.baseVal.value - Vkul.cx.baseVal.value;
    var dY = prz.cy.baseVal.value - Vkul.cy.baseVal.value;
    var d = Math.sqrt((dX*dX)+(dY*dY));
    if (d <= (prz.r.baseVal.value + Vkul.r.baseVal.value)) {
        Vmur.removeChild(prz);
        przeszkody[i] = null;
        if (dX >= 0) Vkul.vh=-Vkul.vh;
        if (dY >= 0) Vkul.vv=-Vkul.vv;
    }
}
if (koniec==true) BudujMur();
}
```

*zmienna koniec* gdy jest prawdziwa, to budujemy mur od nowa, bo wszystkie kulki zostały strącone  
w pętli FOR przeszukujemy całą tablicę przeszkody  
pobieramy kolejną przeszkodę do zmiennej prz  
gdyby była pusta (już strącona) to przerwij wykonywanie instrukcji i wykonaj następny obrót pętli  
gdy jest element to koniec fałsz – jeszcze nie budujemy muru od nowa  
dX i dY przechowują odległość pomiędzy kulą a przeszkodą w pionie i poziomie  
d odległość na płaszczyźnie wyliczona Pitagorasem  
gdy ta odległość jest mniejsza niż promień kuli+promień przeszkody to  
usuwamy z ekranu przeszkodę  
usuwamy przeszkodę z tablicy  
zmieniamy kierunek ruchu kulki

```
Vkul.vv = -Vkul.vv;
}
BudujMur();
```

- Dokument **HTML**, funkcja **BUBLE()** wpisz instrukcje

```
if (ky+kr >= 300) {
    Vkul.vh = 0;
    Vkul.vv = 0;
}
if (ky+kr <= 0){
    Vkul.vv = -Vkul.vv;
}
```

*gdy kula na samym dole* to wyzeruj prędkość kuli – koniec gry  
*gdy kula u samej góry* to odbij w pionie – powtórzone jeszcze raz, bo odbijanie już było

- Zapisz dokumenty i odśwież przeglądarkę
- Zbij wszystkie kulki
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)

