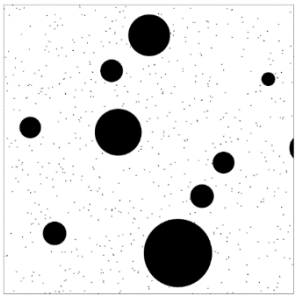
JS 07 Canvas - Laboratorium grawitacyjne (14)



Podstawowego równania grawitacyjnego dostarczył nam Newton ponad 300 lat temu i uczy się go każde dziecko na lekcji fizyki. Obliczenia nie powinny sprawiać większych problemów, gdyż wyliczenia dotyczą przysłowiowego newtonowskiego jabłka. Problem staje się skomplikowany, gdy obliczamy wzajemne oddziaływania dwóch obiektów (np. między pomiędzy Ziemią, a Księżycem), a praktycznie problem staje się nierozwiązywalny bez użycia komputerów, gdy chcemy zbadać tę kwestię dla większej ilości obiektów (np. Układ Słoneczny).
Z matematycznego punktu widzenia zagadnienie nie jest trudne (na poziomie zaawansowanym): należy podwójnie zróżniczkować po czasie równanie grawitacyjne Newtona. Mając masy potrafimy wyliczyć działające siły, z nich przyspieszenia, prędkości i wreszcie położenia obiektów. Otrzymane wyniki, to znane wszystkim tzw. krzywe stożkowe (np. elipsy), po których mogą poruszać się obiekty w przestrzeni. Trzeba też zaznaczyć, że dokładność obliczeń zależy od przyjętego skoku czasowego.

Pamiętaj o tym, by zrzut ekranu DOKUMENTOWAŁ Twoją pracę

Canvas – prostokątna ramka (1)

- W swoim folderze utwórz 2 nowe dokumenty: **js05graw.html js05graw.js**
- Otwórz oba dokumenty w notatniku, a dokument HTML w przeglądarce
- Dokument **HTML,** wklej tekst z ramki

```
<html>
<head>
	<meta charset=utf8>
	<title> GRAWITACJA </title>
	<script src=js05graw.js></script>
</head>
<body>
<center>
	<canvas width=300 height=100 id=GRAW></canvas>
	<br>
	<font size=6>Libront Wacław</font>
</center>
<script>
	var GR=GRAW.getContext("2d");
	var w=GR.canvas.width;
	var h=GR.canvas.height;
	var skok=10;
	var czas;

	animacja();
</script>
</body>
</html>
```

definicja obszaru canvas GR
deklaracja zmiennych: h - wysokość i w – szerokość obszaru canvas
deklaracja zmiennych czas i skok odpowiedzialnych za animację

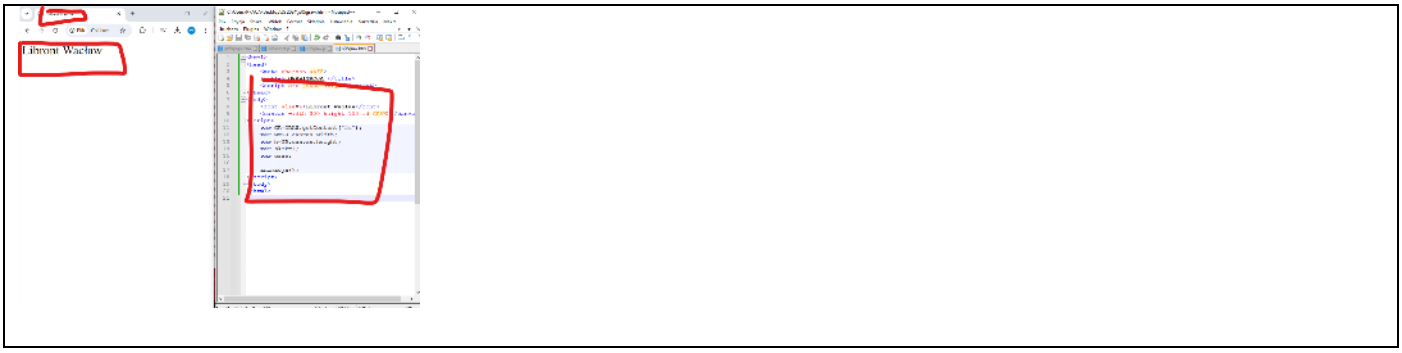
- Dokument **JS**, wklej tekst z ramki

```
function animacja() {
	GR.clearRect(0, 0, w, h);

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

szablon animacji rekurencyjnej

- Zmień tytuł strony **GRAWITACJA** na swoje **inicjały**
- Wpisz swoje **nazwisko i imię**
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)



Ramka (1)

- Dokument **JS**, funkcja **animacja** `clearTimeout(czas);` wpisz instrukcje:

```
GR.strokeStyle="black";
GR.strokeRect(0, 0, w, h);
```

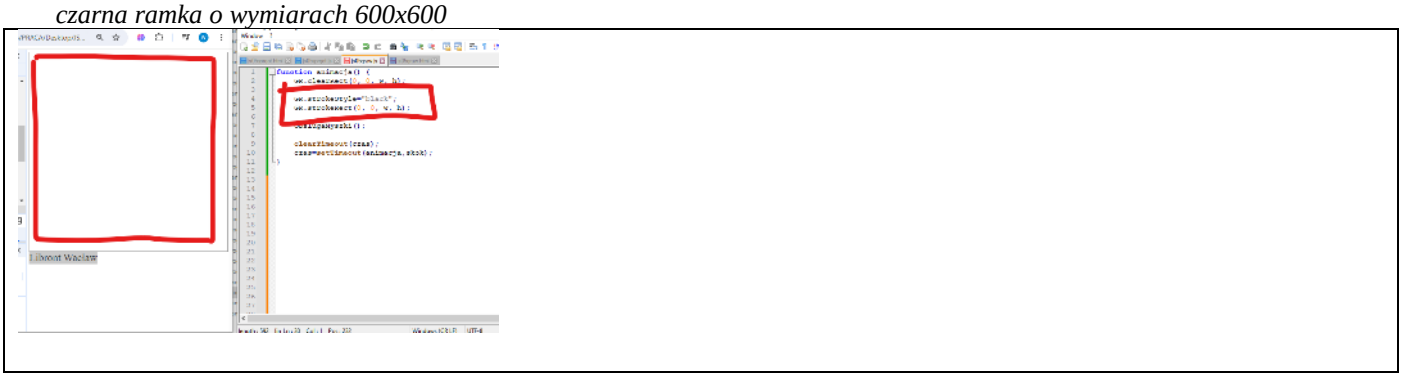
ramka o czarnym brzegu i wymiarach obszaru cavas

- W dokumencie HTML zmień wymiary canvas na **600x600** pikseli

```
<canvas [redacted] id=GRAW></canvas>
```

- Wklej do ramki zrzut ekranu (przeglądarka notatnik)

czarna ramka o wymiarach 600x600



Myszka (1)

Testujemy działanie myszki w obszarze Canvas

- Dokument HTML

```
<font size=6>Libron
</center>
```

wpisz

```
<br>
<label id=TEST> </label>
```

do etykiety wpisujemy położenie myszki podczas przesuwania i klikania

- Dokument **HTML** przed funkcję **animacja()** `animacja();` wklej tekst z ramki

```
var MYSZKA={x:0,y:0};
var MYSZKAon={x:0,y:0};
var MYSZKAoff={x:0,y:0};
var MyszPrzycisk=0;
GR.canvas.onmousemove=function(e) {
       MYSZKA.x=e.offsetX;
       MYSZKA.y=e.offsetY;
};
GR.canvas.onmousedown=function(e) {
       MYSZKAon.x=e.offsetX;
       MYSZKAon.y=e.offsetY;
       MyszPrzycisk=1;
};
GR.canvas.onmouseup=function(e) {
       MYSZKAoff.x=e.offsetX;
       MYSZKAoff.y=e.offsetY;
       MyszPrzycisk=-1;
}
```

funkcja obsługujące przycisk myszki w obszarze canvas

onmousemove *do obiektu MYSZKA wstawiamy współrzędne wskaźnika w trakcie **przesuwania** wskaźnikiem*
onmousedown *do obiektu MYSZKAon wstawiamy współrzędne wskaźnika w momencie **wciśnięcia** przycisku*
onmouseup *do obiektu MYSZKAoff wstawiamy współrzędne wskaźnika w momencie **zwolnienia** przycisku*
współrzędne pobieramy za pomocą własności e.offset
zmienna MyszPrzycisk pełni rolę pomocniczą

- Dokument **JS,** wstaw funkcję testującą myszkę

```
function TestowanieMyszki(){
	document.getElementById('TEST').innerHTML =
	"("+MYSZKA.x+","+MYSZKA.y+") "+
	"("+MYSZKAon.x+","+MYSZKAon.y+") "+
	"("+MYSZKAoff.x+","+MYSZKAoff.y+")";
}
```

- Dokument **JS,** przed `clearTimeout(czas);` wpisz obsługę myszki
  `TestowanieMyszki();`
- Przesuwaj i klikaj myszką po obszarze canvas
- (1) Ustaw wszystkie trzy pola w punkcie (100,100)
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)

(1)

Planeta (1)

- Dokument **HTML** wstaw zmienną `var MyszPrzycisk=0;`
  `var PLANETA={x:0,y:0,vx:0,vy:0,r:10,k:"black"};`

  zm*ienna zapamiętuje podstawowe parametry planety*
  x,y - położenie
  vx,vy - prędkość
  r - promień
  k - kolor
- Dokument **JS** wstaw funkcje rysujące planetę i obsługę współrzędnych

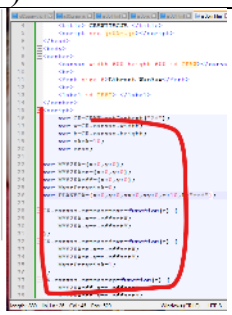
```
function Planeta(x,y,r,k) {
	GR.beginPath();
	GR.strokeStyle=k;
	GR.fillStyle=k;
	GR.arc(x,y,r,0,2*Math.PI);
	GR.stroke();
	GR.fill();
};
function ObslugaMyszki(){
	if (MyszPrzycisk==1){
		PLANETA.x=MYSZKAon.x;
		PLANETA.y=MYSZKAon.y;
	}
	if (MyszPrzycisk==-1){
		MyszPrzycisk=0;
	}
	if (MyszPrzycisk==0){
	}
	Planeta(PLANETA.x,PLANETA.y,PLANETA.r,PLANETA.k)
}
```

ObsługaMyszki
gdy trzymamy wciśnięty przycisk myszki (MyszPrzycisk==1)
przepisujemy zmienne myszki do planety
rysowana jest planeta

gdy puścimy przycisk (MyszPrzycisk==-1)

- Dokument **JS,** funkcja **animacja,** przed `clearTimeout(czas);` wpisz obsługę myszki
  `ObslugaMyszki();`
- Zmień kolor rysowanej planety na **czerwony**
  `var PLANETA={ };`
- (1) Narysuj planetę w dowolnym miejscu
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)

(1)



Rozmiar - Masa (1)

tworzymy suwak, którym ustalimy promień (masę) planety

- Dokument HTML
```
        <label id=TEST> </label>
</center>
```
```
<br>
<input type=range min=1 max=2000 value=100 id=idMAS onchange=FMasa()>
<label id=idMASL> </label>
```
suwak i pole, do którego wpiszemy wartość suwaka za pomocą funkcji Fmasa()

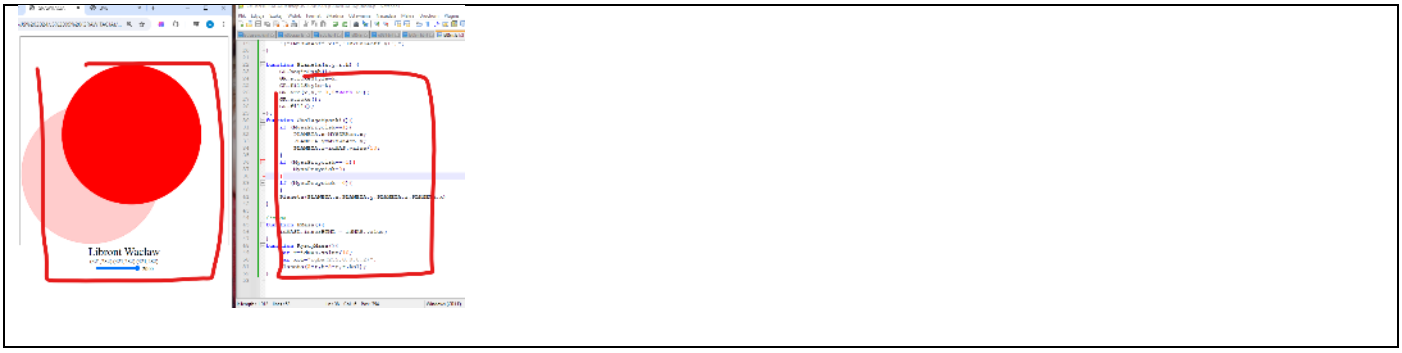
- Dokument HTML
```
animacja();
```
```
FMasa();
```
wartość suwaka od razu widoczna na ekranie

- Dokument JS wklej dwie nowe funkcje

```
function FMasa(){
      idMASL.innerHTML = idMAS.value;
}
function RysujMasa(){
      var r=idMAS.value/10;
      var kol="rgba(255,0,0,0.2)";
      Planeta(5+r,h-5-r,r,kol);
}
```

funkcja Fmasa wstawia wartość suwaka do pola
funkcja RysujMasa rysuje w rogu ekranu czerwone koło symbolizujące ustawianą planetę

- Dokument **JS,** funkcja **animacja,** przed `clearTimeout(czas);`
  `ObslugaMyszki();`
  rysowanie koła masy
```
                PLANETA.y=MYSZKAon.y;
}
```
- Dokument **JS,** funkcja **ObslugaMyszki,**
  `PLANETA.r=idMAS.value/10;`
  promień ustawianej myszką planety jest taki jak ustawiony na suwaku
- Ustaw na suwaku maksymalną wartość
- Kliknij myszką w dowolne miejsce
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)



Laserowa armata (1)

Wciśnięcie i przesunięcie myszki uruchamia „czerwony laserowy dalmierz”
Puszczenie przycisku myszki powoduje wypuszczenie pocisku

- Dokument JS wklej dwie nowe funkcje

```
function LINIA(x1,y1,x2,y2,kol) {
	GR.strokeStyle=kol;
	GR.beginPath();
	GR.moveTo(x1,y1);
	GR.lineTo(x2,y2);
	GR.stroke();
}
function RuchPlanety(){
	PLANETA.x=PLANETA.x+PLANETA.vx;
	PLANETA.y=PLANETA.y+PLANETA.vy;
	Planeta(PLANETA.x,PLANETA.y,PLANETA.r,PLANETA.k)
}
function predkosc(){
	dx=(MYSZKA.x-MYSZKAon.x);
	dy=(MYSZKA.y-MYSZKAon.y);
	var pr=Math.round(Math.sqrt(Math.pow(dx,2)+Math.pow(dy,2)));
	return pr
}
```

LINIA — *rysowanie kresek w określonym kolorze*
RuchPlanety — *planeta rozpoczyna ruch, tam gdzie puszczony przycisk myszki współrzędne ruchu wyznaczają vx i vy*
predkosc — *oblicza prędkość kuli na podstawie długości lasera - tw. Pitagorasa*

- Dokument **HTML**

```
<label id=idLASER> </label>
```

tutaj wpiszemy długość lasera - prędkość wypuszczanej kuli

- Dokument **JS** funkcja **TestowanieMyszki**

```
if (MyszPrzycisk==1) idLASER.innerHTML=predkosc();
```

długość lasera wstawiamy do pola, tylko gdy wciśnięty przycisk

- Dokument **JS**, funkcja **ObslugaMyszki()**

```
if (MyszPrzycisk==1){
	Planeta(MYSZKAon.x,MY
```

wstaw rysowanie czerwonej linii

```
LINIA(MYSZKAon.x,MYSZKAon.y,MYSZKA.x,MYSZKA.y,"red");
```

linia od punktu gdzie wciśnięto przycisk, do punktu w którym znajduje się myszka po przesunięciu

```
if (MyszPrzycisk==-1){
	MyszPrzycisk=0;
```

wstaw obliczanie prędkości

```
PLANETA.vx=(MYSZKAoff.x-MYSZKAon.x)/20;
PLANETA.vy=(MYSZKAoff.y-MYSZKAon.y)/20;
```

na podstawie wielkości przesunięcia obliczamy prędkości pionową i poziomą planety

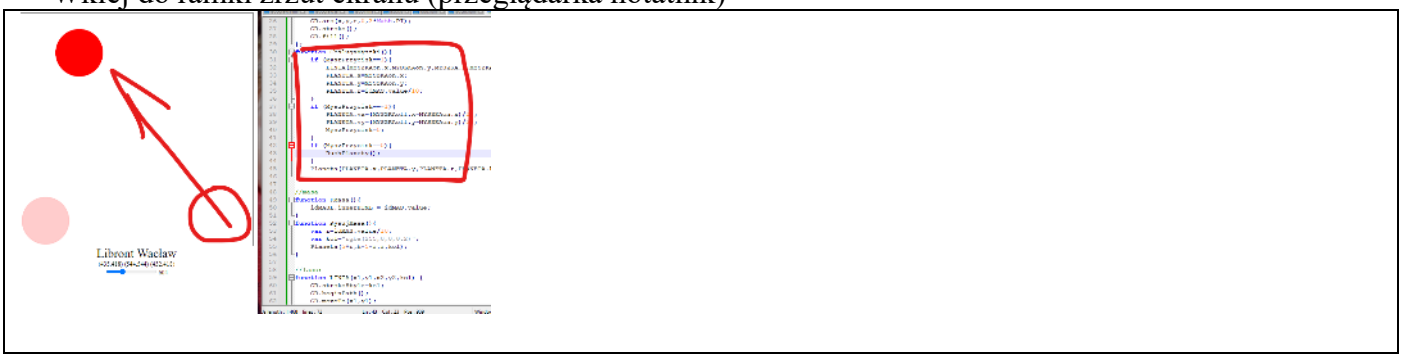
```
if (MyszPrzycisk==0){
}
```

wstaw ruch planety

```
RuchPlanety();
```

na podstawie wielkości przesunięcia obliczamy prędkości pionową i poziomą planety

- Wciśnij przycisk przycisk, przesuń myszkę i wypuść kulę
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)



Odbicia (1)

Rzucona kulka będzie odbijała się od ścian

- Dokument JS, funkcja **RuchPlanety()** wpisz instrukcje odbijania

```
if (PLANETA.x<0 || PLANETA.x>w) PLANETA.vx=-PLANETA.vx;
if (PLANETA.y<0 || PLANETA.y>h) PLANETA.vy=-PLANETA.vy;
```

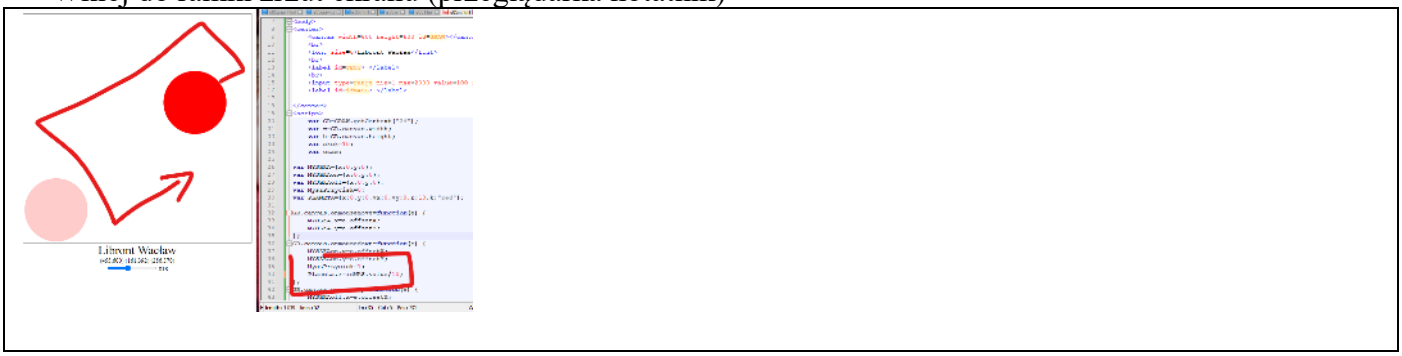
gdy współrzędna planety przekroczy brzeg, to zmień kierunek ruchu na przeciwny

- Dokument HTML, polecenie **GR.canvas.onmousedown** wpisz

```
Planeta.r=idMAS.value/10;
```

wciśnięcie przycisku ustawia wielkość planety zgodną z suwakiem

- Wypuść dużą kulkę po przekątnej i wykonaj zrzuty ekranu, gdy planeta będzie na brzegu
kulka odbija się od brzegów
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)



Rzucamy kulki (1)

Wypuszczamy wiele kulek i wszystkie odbijają się od brzegów. Jest to możliwe, gdy zapamiętamy (w tablicy) wszystkie współrzędne kulek i będziemy rysować wszystkie kuli.

- Dokument **HTML**

```
<label id=idKULE> </label>
```

liczba wypuszczonych kul

- Dokument **HTML**

deklaracja tablicy TP na współrzędne planet

```
var PLANETY=[];
```

- Dokument **JS** funkcja **ObslugaMyszki()** wpisz

```
StartPlanety();
idKULE.innerHTML = PLANETY.length;
```

gdy puszczamy myszkę, to ustawiamy parametry nowej planety i wypisujemy liczbę planet

- Dokument **JS** wklej nową funkcję

```
function StartPlanety(){
	var PL={
		x:PLANETA.x,
		y:PLANETA.y,
		vx:PLANETA.vx,
		vy:PLANETA.vy,
		r:PLANETA.r,
		m:PLANETA.r*10,
		k:"black"
	};
	PLANETY.push(PL);
	PLANETA.r=0;
}
```

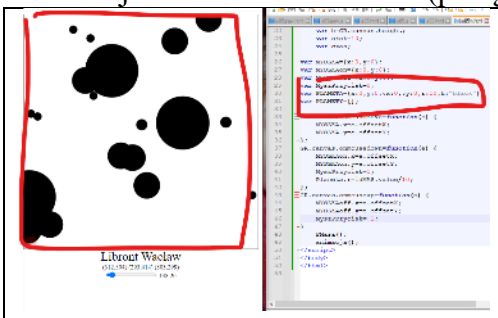
do obiektu PL wstawiamy aktualne parametry myszki-planety a obiekt PL dodajemy do tablicy PLANETY

- Dokument **JS** funkcja **RuchPlanety()**
  - usuń wszystkie instrukcje z funkcji (lub weź w komentarz
  - wklej tekst z ramki

```
for (var i=0;i<PLANETY.length;i++){
	PLANETY[i].x=PLANETY[i].x+PLANETY[i].vx;
	PLANETY[i].y=PLANETY[i].y+PLANETY[i].vy;
	if (PLANETY[i].x<0 || PLANETY[i].x>w) PLANETY[i].vx=-PLANETY[i].vx;
	if (PLANETY[i].y<0 || PLANETY[i].y>h) PLANETY[i].vy=-PLANETY[i].vy;
	Planeta(PLANETY[i].x,PLANETY[i].y,PLANETY[i].r,PLANETY[i].k);
}
```

funkcja RysujPlanety() rysowanie wszystkich planet w tablicy w pętli for
wykonujemy podobne czynności jak poprzednio, ale pobieramy dane z tablicy (i zapisujemy) PLANETY

- Zmień kolor planet na **czarny**
- Wypuść za pomocą myszki mnóstwo planet różnej wielkości
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)



Zderzenia – ruchy Browna (1)

Zderzenia sprężyste, to zderzenia, w których całkowita energia kinetyczna i całkowity pęd układu zderzających się ciał są zachowane. Rozwiązywanie tego typu zadań nie jest trudne i wymaga podstawowych matematyczny umiejętności. Wychodząc od wzorów na energię i pęd można wyliczyć wzory, prędkości ciał po zderzeniu: $V_1' = \frac{m_1 - m_2}{m_1 + m_2} V_1 + \frac{2m_2}{m_1 + m_2} V_2$ $V_2' = \frac{2m_1}{m_1 + m_2} V_1 + \frac{m_2 - m_1}{m_1 + m_2} V_2$
Wystarczy jeszcze dodać, że potrzebne są dwie pętle, bo musimy analizować położenie wszystkich obiektów względem pozostałych.

- Dokument **JS**, funkcja **animacja()** `clearTimeout(czas);` , wpisz tekst `ObliczZderzenia();`

funkcja obliczająca zderzenia

- Dokument **JS**, wklej tekst z ramki

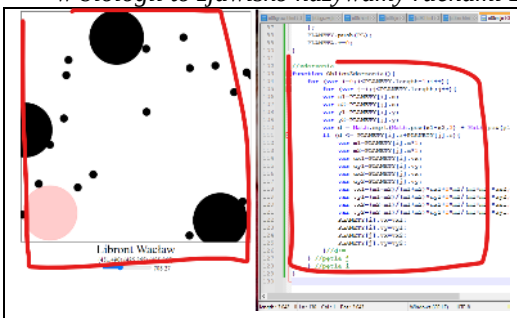
```
function ObliczZderzenia(){
	for (var i=0;i<PLANETY.length-1;i++){
	for (var j=i;j<PLANETY.length;j++){
		var x1=PLANETY[i].x;
		var x2=PLANETY[j].x;
		var y1=PLANETY[i].y;
		var y2=PLANETY[j].y;
		var d = Math.sqrt(Math.pow(x1-x2,2) + Math.pow(y1-y2,2));
		if (d <= PLANETY[i].r+PLANETY[j].r){
```

```
                    var m1=PLANETY[i].m*1;
                    var m2=PLANETY[j].m*1;
                    var ux1=PLANETY[i].vx;
                    var uy1=PLANETY[i].vy;
                    var ux2=PLANETY[j].vx;
                    var uy2=PLANETY[j].vy;
                    var vx1=(m1-m2)/(m1+m2)*ux1+2*m2/(m1+m2)*ux2;
                    var vy1=(m1-m2)/(m1+m2)*uy1+2*m2/(m1+m2)*uy2;
                    var vx2=(m2-m1)/(m1+m2)*ux2+2*m1/(m1+m2)*ux1;
                    var vy2=(m2-m1)/(m1+m2)*uy2+2*m1/(m1+m2)*uy1;
                    PLANETY[i].vx=vx1;
                    PLANETY[i].vy=vy1;
                    PLANETY[j].vx=vx2;
                    PLANETY[j].vy=vy2;
                }//d<=
        } //pętla j
        } //pętla i
}
```

funkcja obliczająca zderzenia
na końcach instrukcji złożonych opisy, co ułatwia orientację w wielopoziomowych funkcjach
pierwsza pętla FOR od 0 do liczby obiektów-1 – działanie ostatniego na ostatni nie analizujemy
droga pętla FOR od elementu „itego" do liczby obiektów - sprawdzamy działanie 1 na 2, to już nie trzeba analizować 2 na 1
pobieramy do zmiennych x1y1x2y2 dane z tablicy - łatwiej zapanować nad skomplikowanymi wzorami
obliczamy odległość d od środków dwóch analizowanych kul
masy kul związane są z ich promieniami
jeżeli odległość d jest mniejsza niż suma ich promieni, to znaczy, że się zderzyły i muszą się odbić
przed obliczenie prędkości po odbiciu (zgodnie z zasadą zachowania energii i pędu) pobieramy do zmiennych dane z tablic
po obliczeniu prędkości wstawiamy je do tablic

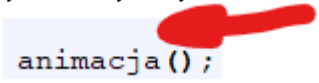
- Wrzuć do układu 20-30 małych ruchomych kulek i 1-2 nieruchome masywne kule
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)

w biologii to zjawisko nazywamy ruchami Browna



Grawitacja – wszystko się przyciąga (1)

W Kosmosie przestrzeń jest nieograniczona i planety uciekają. Rzadko też dochodzi do sprężystych odbić, ale raczej do wchłonięcia małego przez duże, jak większość meteorytów uderzających w Księżyc czy Ziemię. Wyjątkiem jest powstanie Księżyca około 4 miliardy lat temu po tym, jak uderzyło w Ziemię „coś" i wyrwało taki właśnie spory kęs materii. No i najważniejsze – obiekty przyciągają się, i to napędza cały Wszechświat.
Obiekty w Kosmosie nie poruszają się ruchem jednostajnie prostoliniowym, ale zgodnie z teorią Newtona, wszystkie nawzajem się przyciągają. jak przejść od ogólnie znanego wszystkim wzoru na siłę grawitacji do położeń, prędkości i przyspieszeń? Matematycznie nazywamy to różniczkowaniem, a jednym ze sposobów jest metoda Eulera.

- Dokument **HTML** 

wpisz zmienne

```
var dt=0.1;
var G=10;
```

zmienna dt opisuje jak szybko upływa czas = skok czasu
zmienna G jest stałą grawitacji z równania Newtona (przyjmujemy 10 w rzeczywistości jest dużo, dużo mniejsza 6.67e-23)

- Dokument **JS** wklej tekst z ramki

```
function Euler(){
        for (var i=0;i<PLANETY.length;i++){
                PLANETY[i].ax=0;
                PLANETY[i].ay=0;
        for (var j=0;j<PLANETY.length;j++){
                if (i!=j && PLANETY[i].m>0 && PLANETY[j].m>0){
                        var x1=PLANETY[i].x;
                        var x2=PLANETY[j].x;
```

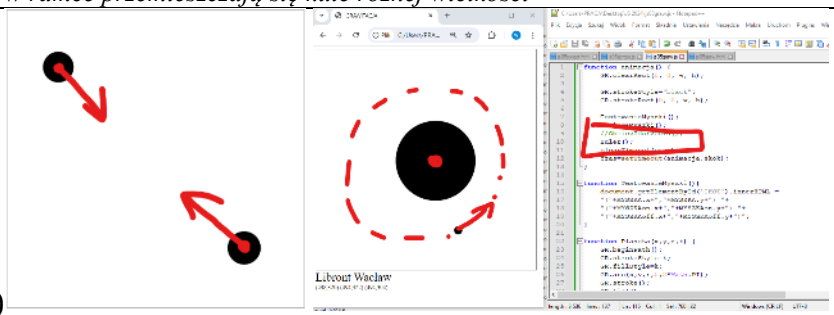
```
                    var y1=PLANETY[i].y;
                    var y2=PLANETY[j].y;
                    var d = Math.sqrt(Math.pow(x1-x2,2) + Math.pow(y1-y2,2));
                    PLANETY[i].ax = PLANETY[i].ax + -G*PLANETY[j].m*(x1-x2)/ Math.pow(d,3);
                    PLANETY[i].ay = PLANETY[i].ay + -G*PLANETY[j].m*(y1-y2)/ Math.pow(d,3);
            } // i!=j
      } // petla j
      PLANETY[i].vx = PLANETY[i].vx + PLANETY[i].ax * dt;
      PLANETY[i].vy = PLANETY[i].vy + PLANETY[i].ay * dt;
      PLANETY[i].x = PLANETY[i].x + PLANETY[i].vx * dt;
      PLANETY[i].y = PLANETY[i].y + PLANETY[i].vy * dt;
      } // petla  i
}
```

funkcja różniczkująca Eulera – z przyspieszeń dochodzimy do prędkości i położeń obiektów
dwie pętle FOR, bo analizujemy wpływa każdego obiektu na każdy inny (prócz samych siebie)
analiza, tylko gdy nie te same obiekty i gdy mają masę większą od zera
pobieramy do zmiennych x1x2y1y2 dane z tablic - łatwiej opanować skomplikowane wzory
obliczamy odległość między planetami
masy są takie same jak promienie planet
z równania na grawitację wyliczamy przyspieszenia i sumujemy je z przyspieszeniami od innych obiektów
po zakończeniu sumowania przyspieszeń działających na jedną planetę, obliczamy nowe prędkości
po obliczeniu prędkości obliczamy nowe położenia (typowe równania ruchu z fizyki do klasy pierwszej)

Dokument **JS,** funkcja **animacja()**,

- usuń funkcję `ObliczZderzenia();`
- wpisz funkcję `Euler();`

- (1) Wstaw dwa nieruchome duże punkty oddalone od siebie
  obiekty kosmiczne zaczną się przyciągać siłami grawitacja
  po zderzeniu układ zaczyna zachowywać się niestabilnie
- Wstaw bardzo duży obiekt na środku (Słońce) i mały obiekt, któremu nadaj niewielką prędkość (Ziemia)
  planeta powinna zacząć okrążać centralną gwiazdę
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)
  w ramce przemieszczają się kule różnej wielkości

(1) 

Wchłanianie (1) – Czarne dziury

Zderzenie w świecie kosmicznym polega w większości przypadków na wchłonięciu małego przez duży. Masy się łączą, a prędkość wypadkowa wyliczona może być podobnie jak przy zwykłych zderzeniach - z zasad zachowania energii i pędu

- Dokument **JS** wklej funkcję z ramki

```
function Zderzenia(d,i,j){
      if (d <= PLANETY[i].r+PLANETY[j].r){
            if (PLANETY[i].m>=PLANETY[j].m){
                  var m2=i;
                  var m1=j;
            } else {
                  var m2=j;
                  var m1=i;
            }
            PLANETY[m2].vx=(PLANETY[m2].vx*PLANETY[m2].m +
PLANETY[m1].vx*PLANETY[m1].m)/(PLANETY[m2].m+PLANETY[m1].m);
            PLANETY[m2].vy=(PLANETY[m2].vy*PLANETY[m2].m +
PLANETY[m1].vy*PLANETY[m1].m)/(PLANETY[m2].m+PLANETY[m1].m);
            PLANETY[m2].m=PLANETY[m2].m*1+PLANETY[m1].m*1;
            PLANETY[m2].r=Math.sqrt(PLANETY[m2].r*PLANETY[m2].r+PLANETY[m1].r*PLANETY[m1].r);
            PLANETY[m1].vx=0;
            PLANETY[m1].vy=0;
            PLANETY[m1].ax=0;
```

```
                PLANETY[m1].ay=0;
                PLANETY[m1].m=0;
                PLANETY[m1].r=0;
        }
}
```

ponieważ funkcja Zderzenia jest na zewnątrz funkcji Euler, dlatego niezbędne dane należy podać jako parametry (d,i,j)
jeżeli odległość od środków planet jest mniejsza niż suma ich promieni, to znaczy, że nastąpiło zderzenie
sprawdzamy, która planeta jest większa i ustawiamy zmienne m1 i m2
obliczamy nowe prędkości wypadkowe z zasady zachowania pędu
sumujemy masy (należy mnożyć, bo JS skleja jak teksty
powiększamy promień nowej planety
zerujemy parametry planety, której już nie ma

```
var d = Math.sqrt(
PLANETY[i].ax = PL
```

- Dokument **JS,** funkcja **Euler()** , wstaw funkcję `Zderzenia(d,i,j);`
- Dokument **JS** wklej funkcję z ramki

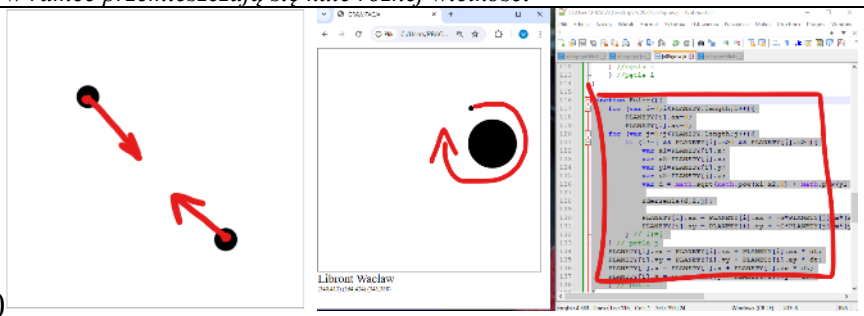
```
function CzyscPlanety(){
        for (var i=0;i<PLANETY.length;i++){
                if (PLANETY[i].m==0) PLANETY.splice(i,1);
        }
}
```

obiekty, które zostały wchłonięte są wyzerowane, ale można jeszcze usunąć je z tablicy TP

```
Euler();
clearTimeout(czas);
```

- Dokument **JS,** funkcja **animacja()** wstaw funkcję `CzyscPlanety();`
- (1) Wstaw dwa nieruchome punkty oddalone od siebie
  zaczną się przyciągać (działa grawitacja) i połącz się w jeden, jeżeli miały takie same masy, to nowy obiekt jest nieruchomy
- Wstaw duży obiekt na środku (Słońce) i mały obiekt (Ziemia), któremu nadaj prędkość
  planeta powinna zacząć okrążać centralną gwiazdę
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)
  w ramce przemieszczają się kule różnej wielkości

(1)



Pierwotny chaos (1)

BUM – czyli Wielki Wychuch, czyli jak z pierwotnego chaosu zrobił się porządek
Ponad 14 miliardów lat temu odbyło wielkie „bum" – i powstał nasz Wszechświat. Na początku jednak była „światłość", którą fizycy zwą fotonami i to z tych najmniejszych cząstek, z tego pierwotnego chaosu narodziły się gwiazdy, a potem planety i życie, które było owocem wybuchających gwiazd.
Wylosowanie 100 maleńkich obiektów będzie próbą symulacji tego pierwotnego chaosu, z którego jednak narodził się porządek, nie dzięki jakimś wielce skomplikowanym procesom, czy też woli „Wielkiego Zegarmistrza", lecz wskutek zwykłego grawitacyjnego przyciągania.
To wszystko, co było na kursie kolizyjnym się połączyło, a cała reszta krąży wokół bardziej masywnych obiektów.

```
<label id=idKULE> </label>
</center>
```

- Dokument **HTML,** wpisz tekst

```
<br>
<input type=range min=1 max=10000 value=1000 id=idBUM onchange=FBum()>
<input type=button value=BUM onclick=LosujPlanety()>
<label id=idILE> </label>
```

suwak do określenia liczby pojawiających się planet po wybuchu

przycisk, którym startujemy Wielki Wybuch

- Dokument **JS** wklej tekst z ramki

```
function losowa(p,k) {
  return Math.floor(Math.random()*(k-p+1)+p);
}
function LosujPlanety(){
       var ile=idBUM.value;
       PLANETY.length=0;
       for (var i=0;i<ile;i++){
              var PL={};
              var mv=10;
              PL.x=losowa(mv,w-mv);
              PL.y=losowa(mv,h-mv);
              PL.vx=losowa(-mv,mv);
              PL.vy=losowa(-mv,mv);
              PL.ax=0;
              PL.ay=0;
              PL.m=losowa(1,3);
              var r=PL.m/10;
              if (r<=0.5){PL.r=0.5} else {PL.r=r;}
              PL.k="black";
              PLANETY.push(PL);
       }
}
function FBum(){
       idILE.innerHTML = idBUM.value;
}
```

funkcja losowe(p,k) losuje liczby całkowite z przedziału <p..k>
funkcja LosujPlanety(ile) wstawia do tablicy TP ILE losowych obiektów, wszystkie parametry są losowe
TP.length=0 najprostszy sposób wyczyszczenia poprzedniej tablicy, aby nie dopisywało nowych w pętli FOR
tworzymy pusty obiekt PL
zmienna mv służy do szybkiego utworzenia granicy dla losowanych danych
promień obliczmy na podstawie masy, gdy jest za mały t powiększamy do 0.5 (1 piksel)

- Dokument **JS**, funkcja **Animacja()**

```
ObslugaMyszki();
Euler();
```

wpisz tekst

```
idKULE.innerHTML = PLANETY.length;
```

liczba planet, które krążą w kosmosie zapisana obok przycisku BUM

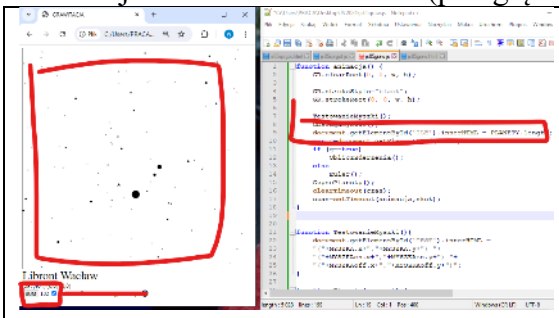
- Dokument **HTML**,

```
animacja();
```

wpisz tekst

```
FBum();
```

- Wciśnij przycisk BUM
  po chwili z pierwotnego chaosu zacznie wyłaniać się kręcący się wokół jednej czarnej dziury porządek
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)



Granice kosmosu (1)

Nasza przestrzeń kosmiczna ograniczona jest krawędziami ramki canvas. Sprawiają to warunki logiczne na zderzenia

- Dokument **HTML**

```
<input type=button val
</center>
```

wpisz tekst

```
<INPUT type=checkbox id=GR checked=true> granice
```

pole opcji, za pomocą którego ustawiamy granice

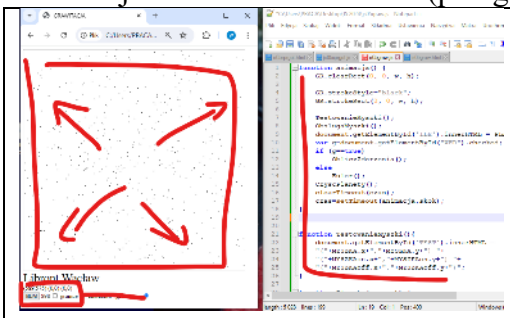
- Dokument **JS**, funkcja **RuchPlanety()** wpisz instrukcję warunkową

zielonym kolorem zaznaczone instrukcje, które już tam są

```
var g=document.getElementById("GR").checked;
if (g==true){
     if (PLANETY[i].x<0 || PLANETY[i].x>w) PLA
     if (PLANETY[i].y<0 || PLANETY[i].y>h) PLA
}
```

pobieramy wartość pola checked
jeżeli jest prawdziwa, to obliczamy zderzenia z granicami ramki

- Zlikwiduj granice ☐ granice
- Wciśnij przycisk BUM
  planety uciekają przez granice - kiedyś powrócą, gdy cięższe obiekty je przyciągną
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)



Laboratorium (1)

Nasze laboratorium może działać jako grawitacyjne (przyciąganie) lub zderzeniowe

- Dokument **HTML**, przed <script> wpisz tekst

```
<INPUT type=checkbox
</center>
```

```
<input type=radio name=wersja id=WER checked=true>
zderzenia - grawitacja
<input type=radio name=wersja>
```

pole opcji, za pomocą którego ustawiamy zderzenia lub grawitację w naszym laboratorium

- Dokument **JS**, funkcja **Animacja()** wpisz instrukcję warunkową
  zielonym kolorem zaznaczone instrukcje, które już tam są

```
var g=document.getElementById("WER").checked;
if (g==true)
     ObliczZderzenia();
else
     Euler();
```

pobieramy wartość pola checked

- Ustaw granice ☐ granice
- Ustaw zderzenia ◉ zderzenia - grawitacja ○
- Wciśnij przycisk BUM
  planety nie uciekają przez granice i cały czas się zderzają
- Wklej do ramki zrzut ekranu (przeglądarka notatnik)