

Laboratorium komputerowe

Projektowanie i tworzenie gier webowych

Część III

„*Podróż statkiem kosmicznym pośród asteroidów*”

## Opis gry.

Nasza gra ma polegać na poruszaniu się statkiem kosmicznym w kosmosie pośród przemieszczających się asteroidów. Zadaniem statku jest unikanie poruszających się asteroid lub ich zestrzeliwanie. Zakładamy, że za każdą zestrzeloną asteroidę otrzymujemy 1 punkt. Podczas manewrowania statkiem może nam przytrafić się kolizja z asteroidą Zakładamy, że na starcie gracz ma trzy życia (trzy utraty statku kosmicznego), a w wyniku kolizji z asteroidą traci 1 życie. Statek kosmiczny powinien poruszać się we wszystkich kierunkach z uwzględnieniem faktu, że nie może wyjść poza obszar przestrzeni kosmicznej (poza obszar animacji).

Dla takich założeń należy zaprojektować grę z użyciem kanw (canvasów), HTML5 i CSS.

## Warto planować!

Aby zaprojektować grę warto zaplanować pewną listę czynności z ustaloną kolejnością. Lista ta pomoże nam uporządkować prowadzone czynności podczas projektowania gry. Dlatego poniżej przedstawiamy propozycję takiej listy zadań.

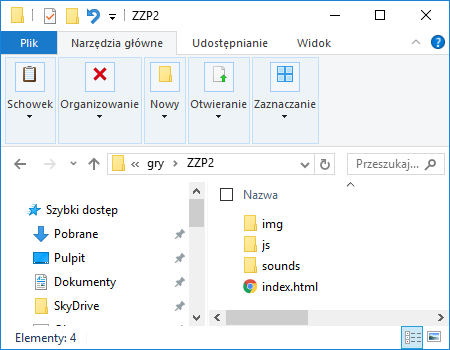
## Lista zadań

1. Struktura katalogów i plików,
2. Specyfikacja elementów gry,
3. Utworzenie szkieletu dokumentu HTML, css oraz pliku javascript,
4. Tworzenie kodu gry:
   1. Obsługa kanw oraz parametryzacja gry,
   2. Przygotowanie głównej animacji – funkcja GlownaGra(),
   3. Tło gry,
   4. Statek kosmiczny
   5. Sterowanie statkiem kosmicznym za pomocą klawiatury,
   6. Pociski statku kosmicznego,
   7. Przemieszczające się asteroidy,
   8. Tablica wyników - liczba punktów, czas gry, liczba żyć,
   9. Czas gry,
   10. Koniec gry – „game over”.
5. Pytania dotyczące gry. Wady i zalety gry.

# Struktura katalogów i plików

Planujemy utworzenie dokumentu HTML (o nazwie index.html), który oprócz arkusza styli ***CSS*** będzie miał dołączony skrypt javascipt o nazwie ***gra.js*** (kod naszej gry) umieszczony z podkatalogu „**js**”. Potrzebne pliki graficzne zostaną umieszczone w podkatalogu „**img”** , a pliki z muzyką/dźwiękami w podkatalogu „**sound**”.

Poniższy rysunek przedstawia zakładaną przez nas strukturę katalogów i plików.



Aby uzyskać taką strukturę katalogów i plików należy pobrać plik **ZZP\_gry\_czesc3.zip** i rozpakować go w odpowiednim miejscu (skąd pobrać ten plik i w jakim miejscu go rozpakować poda prowadzący zajęcia).

# Specyfikacja elementów gry.

Do gry będą potrzebne następujące elementy:

Pliki graficzne umieszczone w podkatalogu **img**.

|  |  |
| --- | --- |
| **Nazwa elementu graficznego** | **Nazwa pliku w katalogu img** |
| Tło animacji gry | tlo800x600.png |
| Statek kosmiczny | statek\_kosmiczny.png |
| Asteroida | asteroida.png |
| Pocisk statku | pocisk.png |

Pliki dźwiękowe umieszczone w podkatalogu **sound**.

|  |  |
| --- | --- |
| **Nazwa elementu graficznego** | **Nazwa pliku w katalogu sound** |
| Wybuch asteroidy | wybuch\_asteroidy.wav |
| Dźwięk strzału statku | strzal\_statku.wav |
| Wybuch statku | wybuch\_statku.wav |

W katalogu **js** w pliku o nazwie **gra.js** zostanie umieszczony kod źródłowy gry napisany w Javascript.

Projektując gry warto rozważyć, aby pewne czynności animacji były wykonywane na różnych kanwach. Dlatego w dokumencie użyjemy trzy warstwy /trzy kanwy/ o identyfikatorach:  
**rezultaty**, **tlo**, **glowny.** Kanwy będą ułożone według odpowiedniej kolejności za pomocą arkusza styli CSS.  
Im wyższa wartość **z-index** tym ta kanwa/warstwa zostaje umieszczona wyżej na stosie warstw.

Rozważamy kanwy o następujących parametrach i ich zastosowaniu:

|  |  |  |  |
| --- | --- | --- | --- |
| **z-index warstwy** | **Id canvas** | **Wielkość obszaru w pikselach** | **Zastosowanie** |
| -3 | rezultaty | 800 x 100 | Wyświetlanie tablicy wyników: punkty, czas gry, liczba żyć |
| -2 | tlo | 800 x 600 | Przewijanie animowanego tła o zdefiniowaną prędkością w pionie |
| -1 | glowny | 800 x 600 | Obszar główny animacji:   * Poruszający się statek kosmiczny * Spadające asteroidy * Pociski statku kosmicznego |

# Utworzenie szkieletu dokumentu HTML, css oraz pliku javascript,

Za pomocą edytora HTML otwórz pusty plik o nazwie index.html i umieść w nim poniższą zawartość dokumentu HTML. Otwórz plik o nazwie **gra.js** z podkatalogu **js**. Zauważ, że w tym pliku nie ma kodu javascript, ale została w nim zawarta zaplanowana struktura komentarzy (poszczególnych sekcji), które będą związane z etapami tworzenia gry.

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="utf-8" />*

*<title> Podróż kosmiczna statkiem pośród asteroidów </title>*

*<style>*

*<!-- Styl CSS -->*

*\* { padding: 0; margin: 0; }*

*canvas {*

*position: absolute;*

*top: 0px;*

*left: 0px;*

*background: transparent;*

*margin: 0 auto;*

*}*

*#rezultaty {*

*z-index: -3;*

*top: 600px;*

*}*

*#tlo {*

*z-index: -2;*

*}*

*#glowny {*

*z-index: -1;*

*}*

*</style>*

*</head>*

*<body>*

*<!-- rezultaty gry -->*

*<canvas id="rezultaty" width="800" height="100">*

*Twoja przeglądarka nie obsługuje elementu Canvas.*

*</canvas>*

*<!-- tlo gry -->*

*<canvas id="tlo" width="800" height="600">*

*Twoja przeglądarka nie obsługuje elementu Canvas.*

*</canvas>*

*<!-- asteroidy, statek, strzały statku -->*

*<canvas id="glowny" width="800" height="600">*

*Twoja przeglądarka nie obsługuje elementu Canvas.*

*</canvas>*

*<!— kod javascipt GRY -->*

*<script src="js/gra.js"></script>*

*</body>*

*</html>*

# Tworzenie kodu gry

Po zapoznaniu się z zawartością pliku ***gra.js*** można łatwo w nim zauważyć, że zostały tam umieszczone odpowiednie sekcje z komentarzami, które umożliwią nam etapowe kodowanie / tworzenie gry.

Dla ułatwienia podajemy opis znaczenia umieszczonych komentarzy:

/\* >>>>>>>>>>>>>> ??? - POCZĄTEK <<<<<<<<<<<< \*/ - sekcja fragmentu kodu mająca początek i koniec

// kod javascript gry

/\* >>>>>>>>>>>>>> ??? - KONIEC <<<<<<<<<<<< \*/

/\* \*\*\*\* początek funkcji/konstruktora \*\*\*\*\* \*/ początek kodu funkcji lub konstruktora

// - komentarz utworzenia zmiennej lub instrukcji

### PRZYKŁADY ODWOŁAŃ DO MIEJSCA W KODZIE:

W celu wskazania umiejscowienia fragmentu kodu gry będziemy posługiwać się skrótem zapisu:

/\* >>>>>>> NAZWA SEKCJI - POCZĄTEK/KONIEC <<<<<<<< \*/ co będzie równoważne z sekcją

/\* >>>>>>> NAZWA SEKCJI – POCZĄTEK <<<<<<<< \*/

// kod gry

/\* >>>>>>> NAZWA SEKCJI - KONIEC <<<<<<<< \*/

Będziemy także umieszczać wywołania funkcji za konkretnym komentarzem np. za komentarzem:

// funkcja rysująca …

***RysujCokolwiek();***

Umieszcza się w ten sposób wywołanie wcześniej zdefiniowanej funkcji o nazwie ***RysujCokolwiek***.

UWAGA!

Kolejne etapy projektowania gry dotyczą działań w pliku **gra.js**   
zlokalizowanym w podkatalogu **js**.

## 4 a) Obsługa kanw oraz parametryzacja gry.

Dostęp do każdej kanwy zrealizujemy za pomocą metody ***document.getElementById***. Kolejnym krokiem będzie ustawienie dla każdej kanwy kontekstu 2d za pomocą metody **getContext**. Poniżej w zaproponowanym kodzie znajduje się propozycja definicji odpowiednich zmiennych z użyciem w/w metod.

Na początku pliku ***gra.js*** za komentarzem // KOD JAVASCRIPT w sekcji komentarza

/\* >> OBSŁUGA CANVAS – POCZĄTEK/KONIEC << /\* , a takie oznaczenie równoważne jest, że jest to sekcja

od /\* >> OBSŁUGA CANVAS – POCZĄTEK << /\* do /\* >>OBSŁUGA CANVAS – KONIEC <<\*/

umieść definicje odpowiednich zmiennych dających dostęp do odpowiednich elementów „canvas”:

// KOD JAVASCRIPT GRY PSPA - Podróż Statkiem Pośród Asteroidów

//

// wersja 0.1

//

/\* >>>>>>>>>>>>>>>>>>>> OBSŁUGA CANVAS - POCZĄTEK <<<<<<<<<<<<<<<<<<<< \*/

// ustalenie zmiennych dostępu do trzech warstw canvas

// odczytanie elementu canvas - rezultaty z dokumentu

***var rezultaty = document.getElementById('rezultaty');***

// zmienna kontekstu canvas dla rezultatów

***var rezctx = rezultaty.getContext('2d');***

// odczytanie elementu canvas - tlo z dokumentu

***var tlogry = document.getElementById('tlo');***

// zmienna kontekstu canvas dla tla

***var tloctx = tlogry.getContext('2d');***

// odczytanie elementu canvas - glowny z dokumentu

***var GlownyCanvas = document.getElementById('glowny');***

// zmienna kontekstu canvas dla glowny

***var glowctx = GlownyCanvas.getContext('2d');***

/\* >>>>>>>>>>>>>>>>>>>> OBSŁUGA CANVAS - KONIEC <<<<<<<<<<<<<<<<<<<< \*/

Projektując grę należy uwzględnić także parametry za pomocą których później można sterować zachowaniem gry. Załóżmy, że za strzelenie jednej asteroidy statek kosmiczny otrzymuje jeden punkt, stąd też wydaje się nam być potrzebny parametr „**punkty**”, który będzie na bieżąco określał liczbę punktów zdobytych przez gracza. Z drugiej strony w przypadku kolizji statku kosmicznego z asteroidą tracimy życie, co oznacza, że jest nam potrzebny parametr „**zycia**”. Możemy też na starcie zdefiniować parametr prędkości pocisków wystrzeliwanych przez statek kosmiczny. Przydatne także wydaje się zliczanie liczby zderzeń z asteroidą, co też implikuje potrzebę parametru „zderzenie”. Z powody tego, że w dalszej części projektu gry planujemy wprowadzić *tak zwany* *timer* odliczający czas co 1 sekundę, więc przewidujemy parametr „**czaszapiscyfrowy**” (zapis czasu w formacie HH:MM:SS). Na koniec dla wielu użytych metod i funkcji może przydać się informacja, czy gra zakończyła się, czy też nie. Dlatego dodatkowo uwzględnimy parametr logiczny (przyjmujący wartość prawdy albo fałszu) o nazwie „**koniec**”. Dla tak opisanych parametrów utworzymy klasę o nazwie ***ParametryGry*** z odpowiednimi składowymi tej klasy. Dodatkowo zdefiniujemy zmienną obiektową klasy ***ParametryGry*** wywołując konstruktor tej klasy z podaniem odpowiednich parametrów. W sekcji /\* >> PARAMETRY GRY - POCZĄTEK / KONIEC << \*/ umieść definicję konstruktora klasy ***ParametryGry*** i odpowiedniej zmiennej według poniższej propozycji:

/\* >>>>>>>>>>>>>>>> PARAMETRY GRY - POCZĄTEK <<<<<<<<<<<<<<<<< \*/

/\* \*\*\*\*\*\*\*\*\*\*\* Konstruktor klasy/obiektu ParametryGry \*\*\*\*\*\*\*\*\*\*\*\* \*/

***function ParametryGry(P, Z, PredPociskow) {***

// punkty - liczba punktów

**this.punkty = P;**

// zycia - liczba żyć / statku kosmicznego

**this.zycia = Z;**

// predkosc\_pociskow - predkosc pocisków wystrzeliwanych przez statek

**this.predkosc\_pociskow = PredPociskow;**

// zderzenie - liczba zderzen statku z asterodiami

***this.zderzenie = 0;***

// zapis cyfrowy czasu gry w formacie HH:MM:SS

***this.czaszapiscyfrowy ="";***

// koniec - wartość prawdy oznacza koniec gry, domyślnie false

***this.koniec = false;***

}

// utworzenie zmiennej obiektowej gra za pomocą konstruktora ParametryGry

// Punkty P = 0, MAKS. LICZBA Żyć Z = 3

// Predkość pocisków: 3 px

***var gra = new ParametryGry(0, 3, 3);***

/\* >>>>>>>>>>>>>> PARAMETRY GRY - KONIEC <<<<<<<<<<<<<<<<<<< \*/

## 4 b) Przygotowanie głównej animacji – funkcja GlownaGra() .

Niech funkcja o nazwie ***GlownaGra()*** będzie dla nas główną funkcją animacji. W kodzie źródłowym tej głównej funkcji zamieścimy odpowiednie komentarze dla wskazania miejsc do których będziemy odwoływać się w kolejnych krokach tworzenia gry. Wewnątrz funkcji **GlownaGra()** została zastosowana instrukcja ***if*** badająca wartość składowej ***koniec*** zmiennej obiektowej ***gra***. Przypominamy, że na starcie wartość ***gra.koniec*** zostaje ustawiona na wartość ***false***. Pod koniec kodu funkcji **GlownaGra()** została użyta metoda ***requestAnimationFrame***, która informuje przeglądarkę o zamiarze wykonania animacji i żąda od przeglądarki ponownego wywołania funkcji ***GlownaGra*** w celu odświeżenia animacji przed następnym odmalowaniem. To właśnie dzięki tej metodzie następuje ciągłe powtarzanie odpowiednich instrukcji i funkcji umieszczonych wewnątrz funkcji ***GlownaGra()***. Przejdź do sekcji /\* > Funkcja GlownaGra – POCZĄTEK/KONIEC < \*/   
i umieść w niej poniższy kod funkcji głównej.

/\* >>>>>>>>>>>>>>>>>>>> Funkcja GlownaGra - POCZĄTEK <<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

// funkcja odpowiedzialna za główna animację Gry

***function GlownaGra()***

**{**

// czyszczenie głownego obszaru canvas

***glowctx.clearRect(0,0,GlownyCanvas.width,GlownyCanvas.height);***

// wywołanie funkcji rysowania tla - przewijanego w pionie

// wywołanie funkcji rysującej rezultaty

// jeśli gra jeszcze się nie skończyła to wykonuj instrukcję z instrukcji if

***if (!gra.koniec)***

***{***

// wywołanie funkcji rysowania statku kosmicznego w bieżącej pozycji

// obsługa sterowania statkiem za pomocą klawiatury

// wywołanie funkcji rysowania pocisków statku

// wywołanie funkcji rysowania asteroid

// informuje przeglądarkę o zamiarze wykonania animacji i żąda od   
 // przeglądarki wywołania określonej funkcji GlownaGra w celu odświeżenia animacji   
 ***requestAnimationFrame(GlownaGra);***

***} else {***

// wywołaj funkcję z informacją, że nastąpił koniec gry

***}***

***}***

/\* >>>>>>>>>>>>>>>>>>> Funkcja GlownaGra - KONIEC <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

Następnie w sekcji /\* \*\*\*\*\* PROGRAM GŁÓWNY GRY \*\*\*\*\*\*\*\*\*\* \*/ za wierszem komentarza

// wywołaj główną animację GlownaGra umieść wywołanie funkcji:

***GlownaGra();***

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PROGRAM GŁÓWNY GRY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// Uruchom timer co 1 sekundę z przeliczeniem czasu gry na zapis cyfrowy,

// Utwórz nową asteroidę co 1000 ms

// wywołaj główną animację GlownaGra

***GlownaGra();***

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* KONIEC PROGRAMU GŁÓWNEGO GRY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

## 4 c) Tło gry.

Wiele gier ma często statyczne tło lub na tle statycznego tła ruszają się inne elementy animacji. W tym rozdziale naszym zadaniem będzie przewijanie tej samej klatki obrazu o rozdzielczości 800 x 600 pikseli jedna za drugą w pionie z zadaną prędkością w pikselach. W celu narysowania tła niezbędna jest definicja zmiennej ***tlo\_grafika*** klasy **Image** oraz wskazania źródła tej grafiki, czyli pliku ***tlo800x600.png***. Na potrzeby narysowania tła został utworzony konstruktor klasy ***ParametryTla*** o składowych ***X, Y*** oraz ***Predkosc***. Została też utworzona zmienna ***tlo*** klasy ***ParametryTla*** o zadanych wartościach pozycji X i Y: (0, 0) oraz prędkości 1 px. Zaprojektowaliśmy metodę ***RysujTlo()***, której zadaniem jest zmiana współrzędnej ***Y*** o wielkość prędkości. W przypadku animacji jeśli współrzędna Y osiągnie lub przekroczy wartość wysokości kanwy ***tlogry.height*** (patrz punkt 4a) to należy ją wyzerować.

W sekcji /\* >>>>> TŁO ANIMACJI – POCZATĘK/KONIEC <<<<< \*/ umieść poniższy kod definiujący zmienną tlo\_grafika, klasy ParametryTla oraz funkcji RysujTlo().

/\* >>>>>>>>>>>>>> TŁO ANIMACJI - POCZATĘK <<<<<<<<<<<<<<<<<<<<<<<<< \*/

// zmienna dla tła grafiki

***var tlo\_grafika = new Image();***

// ustalenie źródła dla tła -> grafika 800 x 600

// wskazanie źródła grafiki tła

***tlo\_grafika.src = "img/tlo800x600.png";***

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Konstruktor klasy ParametryTla \*\*\*\*\*\*\*\*\* \*/

/\* konstruktor klasy ParametryTla \*\*\*\*\* \*/

***function ParametryTla(x, y, p) {***

// X,Y wsp. startowe dla tła

***this.X = x;***

***this.Y = y;***

// Predkosc - predkosc przewijania tła w pionie \*/

***this.Predkosc = p;***

}

// utworzenie zmiennej obiektowej tlo za pomocą konstruktora ParametryTla tła

// pozycja (x,y) = (0,0), a prędkość p = 1px

***var tlo = new ParametryTla(0, 0, 1);***

/\* \*\*\*\*\*\*\*\*\* funkcja RysujTlo - narysowanie / przewijanego tła \*/

***function RysujTlo()***

***{***

// zmien pozycje Y tla o prędkość przewijania

***tlo.Y += tlo.Predkosc;***

// narysuj tlo w pozycji X, Y

***tloctx.drawImage(tlo\_grafika, tlo.X, tlo.Y);***

// narysuj drugie tlo w pozycji Y - wysokość tła gry

***tloctx.drawImage(tlo\_grafika, tlo.X, tlo.Y - tlogry.height);***

// wyzeruj dla tla Y jeśli Y przekroczy wysokość tła grafiki

***if (tlo.Y >= tlogry.height)***

***tlo.Y = 0;***

***}***

/\* >>>>>>>>>>>>>>>>> TŁO ANIMACJI - KONIEC <<<<<<<<<<<<<<<<<<<<<<<<< \*/

### Zmiany w programie głównym

Pozostało jeszcze w funkcji ***GlownaGra()*** za linią komentarza:

// wywołanie funkcji rysowania tła - przewijanego w pionie   
należy dopisać kod wywołania funkcji:

***RysujTlo();***

Otwórz w przeglądarce Chrome plik ***index.html*** i sprawdź działanie gry. Czy animacja przewijanego tła działa? Odśwież jeszcze raz grę za pomocą klawisza F5 i zweryfikuj działanie gry.

## 4 d) Statek kosmiczny

Kolejnym zadaniem jest dołożenie do animacji poruszającego się statku kosmicznego. W chwili obecnej zajmiemy się tylko jego narysowaniem w tak zwanej pozycji startowej. Do tego będzie nam potrzebna zmienna **statek\_grafika** dająca dostęp do grafiki statku, dwie zmienne ***wysStatku, szerStatku*** potrzebne do określenia obszaru zajmowanego przez statek. Zaprojektowany też został konstruktor klasy **Statek** ze składowymi takimi jak: pozycja ***X, Y*** statku, szerokość ***W*** i wysokość ***H*** statku, przyjmująca wartości prawdy lub fałszu składowa ***CzyZderzenie*** będąca statusem informującym, czy statek zderzył się z asteroidą, składowa ***DzwiekZderzenia*** oraz ***DzwiekZderzenia.volume*** są odpowiedzialne za dźwięk wybuch statku, w przypadku jego zderzenia z asteroidą. Dodatkowo potrzebujemy zmienną obiektową ***statek*** utworzoną za pomocą konstruktora ***Statek***. Przypominamy że rysowanie statku kosmicznego odbywa się na kanwie ***glowny*** i dostęp do tego, aby rysować na tej warstwie możliwy jest za pomocą zmiennej ***glowctx***. Zadaniem funkcji o nazwie ***RysujStatekKosmiczny()*** jest rysowanie statku w bieżącej pozycji na kanwie ***glowny***. W przypadku kiedy nastąpi kolizja z asteroidą (analiza tego przypadku odbędzie się w podrozdziale dotyczącym przemieszczania się asteroid) to składowa ***CzyZderzenie*** uzyskuje chwilowo wartość ***True***.

W sekcji /\* >>STATEK KOSMICZNY – POCZĄTEK/KONIEC << \*/ umieść poniższy kod:

/\* >>>>>>>>>>>>>>>>> STATEK KOSMICZNY <<<<<<<<<<<<<<<<<<<<< \*/

// statek kosmiczny grafika

***var statek\_grafika = new Image();***

***statek\_grafika.src = "img/statek\_kosmiczny.png";***

// wysokość i szerokość statku

***var wysStatku = 100;***

***var szerStatku =60;***

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Konstruktor klasy Statek \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

***function Statek(x, y, w, h, Zderzenie)***

***{***

// X, Y współrzędne pozycji statku

***this.X = x;***

***this.Y = y;***

//szerokość statku

***this.W = w;***

// wysokość statku

***this.H = h;***

// status, czy statek zderzył się z asteroidą

***this.CzyZderzenie = Zderzenie;***

// dźwięk odtworzony w przypadku zderzenia z asteroidą

***this.DzwiekZderzenia = new Audio("sounds/wybuch\_statku.wav");***

// głośność dźwięku

***this.DzwiekZderzenia.volume = .22;***

***}***

// utworzenie zmiennej obiektowej statek za pomocą konstruktora klasy Statek

***var statek = new Statek(GlownyCanvas.width/2 - szerStatku/2, GlownyCanvas.height-wysStatku, szerStatku, wysStatku, false);***

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* funkcja RysujStatekKosmiczny \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

***function RysujStatekKosmiczny()***

***{***

// jeśli statek nie zderzył się to narysuj statek w bieżącej pozycji X,Y

***if (!statek.CzyZderzenie)***

***glowctx.drawImage(statek\_grafika, statek.X, statek.Y)***

***else***

***{***

// w przypadku zderzenia z asteroidą ustaw wsp. X,Y

// ponownie jako startowe

***statek.X = GlownyCanvas.width/2- szerStatku/2;***

***statek.Y = GlownyCanvas.height-wysStatku;***

// ustaw status CzyZderzenie na false

***statek.CzyZderzenie = false;***

***}***

***}***

/\* >>>>>>>>>>>>> KONIEC STATEK KOSMICZNY <<<<<<<<<<<<<<<<<<<<< \*/

### Zmiany w funkcji GlownGra()

W celu zobaczenia nieruchomego jeszcze statku w pozycji startowej należy w funkcji ***GlownaGra()*** (wewnątrz instrukcji ***if (!gra.koniec)*** za komentarzem:

// wywołanie funkcji rysowania statku kosmicznego w bieżącej pozycji

// umieścić wywołanie funkcji rysowania statku kosmicznego tzn.

***RysujStatekKosmiczny();***

## 4 e) Sterowanie statkiem kosmicznym za pomocą klawiatury.

Do realizacji sterowania za pomocą klawiatury będą nam potrzebne zmienne logiczne przechowujące stany, czy dany klawisz został naciśnięty, czy też nie. Proponujemy cztery zmienne rightPressed, leftPressed, upPressed, downPressed, które zostały zdefiniowane poniżej z domyślną wartością false, a nazwy ich symbolizują odpowiednie klawisze na klawiaturze. Oprócz tego do kodu gry należy dołożyć obsługę zdarzeń “keydown” oraz “keyup” za pomocą metody ***document. addEventListener*** z odpowiednimi argumentami wywołania tej metody. Zaprojektowane funkcje ***keyDownHandler*** oraz ***keyUpHandler*** mają za zadanie z wykorzystaniem instrukcji switch ustalić, czy dany kod konkretnego klawisza (e.keyCode) został naciśnięty w dół lub zwolniony “up”. Dla wartości e.keyCode = 32, czyli naciśnięcia w dół klawisza spacji zostawiliśmy komentarz z przygotowaniem wywołania funkcji strzału statku, którą opracujemy w następnym rozdziale 4f). Dodatkowo należy zaprojektować funkcję ***SterowanieStatkiemZaPomocaKlawiatury***, której zadaniem jest analiza stanu odpowiednich klawiszy, a przez to też zmiana pozycji ***X, Y*** statku. Funkcja ta pilnuje nasz statek aby nie wyszedł poza obszar kanwy animacji – jest to możliwe dzięki analizie, czy współrzędna ***X*** statku nie osiągnęła pozycji szerokości kanwy odjąć szerokość statku oraz minus ***5 px*** ( tzn. ***GlownyCanvas.width-statek.W-5*** ) i podobnie dla Y tylko dotyczy to wysokości kanwy i statku ( ***GlownyCanvas.height-statek.H-5*** ).

Umieść poniższy kod w sekcji /\* STEROWANIE STATKIEM – POCZĄTEK/KONIEC \*/

/\* >>>>>>>>>>>>>>>>>> STEROWANIE STATKIEM - POCZĄTEK <<<<<<<<<<<<<<<<<<<< \*/

// zmienne globalne - obsługa klawiszy!

// strzalka w prawo

***var rightPressed = false;***

// strzalka w lewo

***var leftPressed = false;***

// strzałka w górę

***var upPressed = false;***

// strzałka w dół

***var downPressed = false***;

/\* /////////// OBSŁUGA ZDARZEŃ ////////////// \*/

// dodanie wywołania metod dla zdarzeń klawiatury - key down, key up

***document.addEventListener("keydown", keyDownHandler, false);***

***document.addEventListener("keyup", keyUpHandler, false);***

/\* funkcja obsługi naciśnięcia klawiszy w dół \*/

***function keyDownHandler(e) {***

***switch(e.keyCode) {***

***case 32:***

// klawisz spacji

// wystrzel pociski

// tutaj w dalszym etapie 4f

***break;***

***case 37:***

// lewa strzalka

***leftPressed = true;***

***break;***

***case 38:***

// strzalka do góry

***upPressed = true;***

***break;***

***case 39:***

// strzałka w prawo

***rightPressed = true;***

***break;***

***case 40:***

// strzałka w dół

***downPressed = true;***

***break;***

***default:***

// nic nie rób

}

}

/\* funkcja obsługi naciśnięcia klawiszy w górę - zwolnienie klawisza \*/

***function keyUpHandler(e) {***

***switch(e.keyCode) {***

***case 37:***

***leftPressed = false;***

***break;***

***case 38:***

***upPressed = false;***

***break;***

***case 39:***

***rightPressed = false;***

***break;***

***case 40:***

***downPressed = false;***

***break;***

***default:***

// nic nie rób

***}***

**}**

/\* ///////////////// Funkcja sterowania statkiem za pomocą klawiatury ///////////// \*/

// zadaniem funkcji oprócz kierowaniem statku góra / dół, lewo - prawo

// jest pilnowanie, aby statek nie wyleciał poza obszar głównego canvasu

***function SterowanieStatkiemZaPomocaKlawiatury()***

***{***

***if(rightPressed && statek.X < GlownyCanvas.width-statek.W-5) {***

***statek.X += 7;***

***}***

***else if(leftPressed && statek.X > 0) {***

***statek.X -= 7;***

***} else if(upPressed && statek.Y > 0) {***

***statek.Y -= 5;***

***}***

***else if(downPressed && statek.Y < GlownyCanvas.height-statek.H-5)***

***{***

***statek.Y += 5;***

***}***

***}***

/\* >>>>>>>>>>>>>>>>>> STEROWANIE STATKIEM - KONIEC <<<<<<<<<<<<<<<<<<<< \*/

### Zmiany w funkcji GlownGra()

W celu uaktywnienia sterowania statkiem należy w funkcji ***GlownaGra()*** (wewnątrz instrukcji ***if (!gra.koniec)*** za komentarzem:

// obsługa sterowania statkiem za pomocą klawiatury

// umieścić wywołanie funkcji sterowania statkiem, tzn.

***SterowanieStatkiemZaPomocaKlawiatury();***

Otwórz plik index.html w przeglądarce Chrome i sprawdź działanie poruszania się statku!

## 4 f) Pociski statku kosmicznego

Naszym głównym założeniem jest to, że statek podczas strzału wystrzeliwuje równolegle dwa pociski z lewego i prawego działa (zobacz poniżej w kodzie źródłowym funkcji ***WystrzeleniePociskow***). Może zdarzyć się, że jeden z nich trafi do celu, zaś drugi może opuścić obszar animacji (wówczas będzie zniszczony jako obiekt). Definiujemy zmienną ***pocisk*** dającą dostęp do grafiki ***pocisk.png***. Potrzebna też nam będzie tablica pocisków przechowująca obiekty klasy ***Pocisk***. Sama klasa ***Pocisk*** powinna zawierać współrzędną pocisku, czy trafił w cel (w asteroidę), prędkość pocisku oraz czy ma zostać odtworzony dźwięk strzału w momencie wystrzeliwania pocisków. Jedną z ważniejszych funkcji jest **RysujPociski(),** której zadaniem jest rysowanie pocisków, weryfikacja, czy pocisk trafił w cel i oznaczenie go, aby nie był już rysowany w dalszym działaniu gry.

Funkcja ***RysujPociski()*** uwzględnia użycie funkcji **DetekcjaKolizjiPociskowZAsterodiami(i)**, której dołączono tylko szkielet z tego powodu, gdyż nie jest gotowy etap projektowania gry z użyciem asteroidów.

Umieść poniższy kod w sekcji /\* >>> POCISKI - POCZĄTEK/KONIEC<<<< \*/

/\* >>>>>>>>>>>>>>>>>>>>>>>>> POCISKI - POCZĄTEK <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

// pociski / laserowe statku kosmicznego

// grafika pocisku

***var pocisk = new Image();***

***pocisk.src = "img/pocisk.png";***

// Zmienne tablicowa przechowująca obiekty klasy pocisk

***var pociski = new Array();***

// Konstruktor klasy Pocisk

***function Pocisk(x, y, p, dzwiek\_strzalu) {***

// współrzędna X

***this.X = x;***

// współrzędna Y

***this.Y = y;***

// informacja czy pocisk trafił w asteroidę

***this.CzyTrafilCel = false;***

// prędkość pocisku

***this.Predkosc = p;***

// czy odtworzyć dźwięk w chwili strzału

***this.CzyDzwiekStrzalu = dzwiek\_strzalu;***

// dzwięk strzału

***this.DzwiekStrzalu = new Audio("sounds/strzal\_statku.wav");***

// głośność strzału

***this.DzwiekStrzalu.volume = .12;***

***}***

/\* //////////////////// Funkcja WystrzeleniePociskow //////////////////////////// \*/

***function WystrzeleniePociskow() {***

// umieść w tablicy pocisk lewy

// umieść tablicy pocisków pocisk lewy z odtwórz dźwięk wystrzału dla pocisku lewego

***pociski.push(new Pocisk(statek.X + szerStatku/2 - 20, statek.Y+wysStatku/2, gra.predkosc\_pociskow, true));***

// umieść w tablicy pocisk prawy

***pociski.push(new Pocisk(statek.X + szerStatku/2 + 20, statek.Y+wysStatku/2, gra.predkosc\_pociskow, false));***

***}***

// Funkcja rysująca pociski statku kosmicznego

***function RysujPociski() {***

// Usuń pociski z listy te, które trafiły do celu lub wyszły poza obszar animacji

***for (var i = 0; i < pociski.length; i++)***

***{***

***if (pociski[i].CzyTrafilCel == true || pociski[i].Y <= 0) {***

// usuń pocisk z tablicy

***pociski.splice(i, 1);***

***i--;***

***}***

***}***

// Rysowanie pocisków oraz

// weryfikacja, czy nastąpiło uderzenie pocisku w asteroidę

***for (var i = 0; i < pociski.length; i++) {***

// zmień wsp. Y danego pocisku o jego prędkość

***pociski[i].Y -= pociski[i].Predkosc;***

// Sprawdź, czy pocisk trafił w asteroidę

// detekcja kolizji pocisków z asteroidami

***DetekcjaKolizjiPociskowZAsterodiami(i);***

// rysuj pocisk jeśli nie trafił asteroidy

***if (!pociski[i].CzyTrafilCel)***

***{***

***glowctx.drawImage(pocisk, pociski[i].X, pociski[i].Y);***

// jeśli to nowy pocisk to odtwórz dźwięk strzału

***if (pociski[i].CzyDzwiekStrzalu == true)***

***{***

***pociski[i].DzwiekStrzalu.play();***

***pociski[i].CzyDzwiekStrzalu = false;***

***}***

***}***

***}***

***}***

// Funkcja dokonująca detekcji kolizji pocisków z asteroidami

***function DetekcjaKolizjiPociskowZAsterodiami(i)***

***{***

// kod będzie uzupełniony później

***}***

/\* >>>>>>>>>>>>>>>>>>>>>>>>> POCISKI - KONIEC <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

### Zmiany w innych miejscach kodu

W celu uaktywnienia rysowania pocisków należy w funkcji ***GlownaGra()*** (wewnątrz instrukcji ***if (!gra.koniec)*** za komentarzem:

// wywołanie funkcji rysowania pocisków statku umieść wiersz

***RysujPociski();***

Oprócz tego wróć do kodu funkcji sterowania klawiaturą ***keyDownHandler(e)***, w której dla obsługi klawisza spacji (e.keyCode = 32) za komentarzem // wystrzel pociski usuń komentarz //tutaj w dalszym etapie 4f i zastąp ten wiersz wywołaniem funkcji:

***WystrzeleniePociskow();***

Otwórz plik index.html w przeglądarce Chrome, użyj klawisza spacji i sprawdź działanie strzałów statku.

## 4 g) Przemieszczające się asteroidy.

Kolejnym zadaniem jest uzupełnienie gry o przemieszczające się asteroid. W tym celu będzie nam potrzebna zmienna dająca dostęp do grafiki asteroidy oraz odpowiednia klasa definiująca asteroidę. Należy także rozważyć oprócz funkcji rysowania asteroid funkcję detekcji kolizji asteroidy ze statkiem. Jak już będziemy gotowi z asteroidami to będziemy mogli wrócić i uzupełnić kod funkcji ***DetekcjaKolizjiPociskowZAsterodiami*** z rozdziału ***4f***. Do realizacji spadających asteroidów będzie nam potrzebny obiekt asteroida.

Definiujemy zmienna graficzną ***asteroida*** ze źródłem ***asteroida.png*** oraz zmienną tablicową ***asteroidy*** przechowującą asteroidy typu obiektowego **Asteroida**. Obiekt asteroida będzie używał takich składowych jak współrzędne X, Y, prędkość asteroidy w poziomie i pionie, informacja czy asteroida została zniszczona oraz odpowiednie dźwięki związane z wybuchem asteroidy.

Ważnym aspektem jest generowanie kolejnych asteroid na podczas rozgrywki. W związku z tym proponujemy funkcję **UtworzAsteroide**, której zadaniem jest umieszczanie w tablicy ***asteroidy*** nową asteroidę z wylosowaną: pozycją **X i Y**, prędkością. Jednak tą funkcję użyjemy w programie głównym za pomocą metody ***setInterval*** z interwałem na starcie ***1 sekunda (1000 ms)***.

Z drugiej strony proponowana funkcja ***RysujAsteroidy()*** musi uwzględniać usuwanie asteroid z tablicy obiektów, w przypadku kiedy została ona zestrzelona oraz dokonywać detekcji kolizji ze statkiem (to znaczy na ile statek zbliżył się do asteroidy). W przypadku kolizji asteroidy następuje aktualizacja wyników gry w sekcji   
**/\* AKTUALIZACJA WYNIKÓW GRY \*/**. Oprócz aktualizacji wyników wywoływana jest także odpowiednia funkcja ***ZderzenieStatkuzAsteroida()***, której zadaniem jest wyzerowanie tablicy pocisków, asteroidów po to, aby statek kosmiczny mógł wrócić do pozycji początkowej, a gracz będzie dalej kontynuował grę.

Umieść poniższy kod w sekcji /\* >>ASTEROIDY – POCZĄTEK/KONIEC << \*/

/\* >>>>>>>>>>>>>>>>>>>>>>>>>>> ASTEROIDY - POCZĄTEK <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

// asteroida - element graficzny

***var asteroida = new Image();***

***asteroida.src = "img/asteroida.png";***

// Zmienna tablicowa przechowujące informacje o obiektach klasy Asteroida

***var asteroidy = new Array();***

// konstruktor klasy Asteroida

***var Asteroida = function (x, y, s, g)***

***{***

// współrzędna X asteroidy

***this.X = x;***

// współrzędna Y asteroidy

***this.Y = y;***

// prekosc w poziomie

***this.PredkoscWPoziomie = s;***

// predkość w pionie / spadania asteroidy

***this.PredkoscWPionie = g;***

// status, czy asteroida została zniszczona

***this.CzyZniszczona = false;***

// źródło dzwięku eksplozji asteroidy

***this.DzwiekWybuchuAsteroidy = new Audio("sounds/wybuch\_asteroidy.wav");***

// głośność dźwięku

***this.DzwiekWybuchuAsteroidy.volume = .22;***

***}***

/\* \\\\\\\\\\\\\\\\\\\ funkcja tworząca asteroidę ///////////////////// \*/

***function UtworzAsteroide()***

***{***

// wyslosowanie prędkości asteroidy

**var los\_PredkoscWPoziomie = Math.round(Math.random() \* 2)+1;**

***var los\_PredkoscWPionie = Math.round(Math.random() \* 2)+1;***

***var los\_X= Math.round(Math.random() \* (GlownyCanvas.width - asteroida.width));***

***var los\_Y= 10 + Math.random() \* 20;***

// zmiana kierunku z prawej do lewej

***if (los\_X > 0.5 \* GlownyCanvas.width) { los\_PredkoscWPoziomie = -los\_PredkoscWPoziomie; }***

***asteroidy.push(new Asteroida(los\_X, los\_Y, los\_PredkoscWPoziomie, los\_PredkoscWPionie));***

***}***

/\* \\\\\\\\\\\\\\\\\ Funkcja rysująca asteroidy ///////////////// \*/

***function RysujAsteroidy() {***

// Usuń asteroidę z ekranu jeśli została zestrzelona przez statek kosmiczny

***for (var i = 0; i < asteroidy.length; i++) {***

***if (asteroidy[i].CzyZniszczona == true || asteroidy[i].X > GlownyCanvas.width || asteroidy[i].Y > GlownyCanvas.height)***

***{***

***asteroidy.splice(i, 1);***

***i--;***

***}***

***}***

// Narysuj asteroidę, która nie została zestrzelona

***for (var i = 0; i < asteroidy.length; i++)***

***{***

// detekcja kolizji asteroidy ze statkiem

***DetekcjaKolizjiAsteroidyZeStatkiem(i);***

// jeśli asteroida nie jest zniszczona to zmień jej pozycję oraz narysuj ją

***if (!asteroidy[i].CzyZniszczona)***

***{***

// zmien pozycję X asteroidy o PredkoscWPoziomie

***asteroidy[i].X += asteroidy[i].PredkoscWPoziomie;***

// zmien pozycję Y asteroidy o PredkoscWPionie

***asteroidy[i].Y += asteroidy[i].PredkoscWPionie;***

// narysuj asteroidę w nowej pozycji

***glowctx.drawImage(asteroida, asteroidy[i].X, asteroidy[i].Y);***

}

***}***

// jeśli nastapiło zderzenie statku z asteroidą to wywołaj funkcję

// ZderzenieStatkuzAsteroida

***if (statek.CzyZderzenie) ZderzenieStatkuzAsteroida();***

***}***

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* funkcja detekcji kolizji statku z asteroidami \*\*\*\*\*\*\* \*/

***function DetekcjaKolizjiAsteroidyZeStatkiem(i)***

***{***

// detekcja kolizji asteroidy ze statkiem

***if ((Math.abs(asteroidy[i].X - statek.X) < 35) && (Math.abs(asteroidy[i].Y - statek.Y) < 32)) {***

// oznacz, że statek uległ uszkodzeniu

***statek.CzyZderzenie = true;***

// odtworz dzwiek zderzenia statku z asteroidą

***statek.DzwiekZderzenia.play();***

// oznacz asteroidę jako zniszczoną

***asteroidy[i].CzyZniszczona = true;***

/\* ///// AKTUALIZACJA WYNIKÓW GRY ////// \*/

// zmniejsz liczbę żyć o 1

***gra.zycia--;***

// zwiększ liczbę zderzeń statku o 1

***gra.zderzenie++;***

// jeśli liczba żyć wynosi zero to koniec gry

***if (gra.zycia == 0 )***

***{***

***gra.koniec= true;***

***}***

/\* /////// KONIEC AKTUALIZACJAI WYNIKÓW GRY ////// \*/

***}***

***}***

/\* Funkcja ZderzenieStatkuzAsteroida \*/

// zadaniem funkcji jest wyzerowanie

// tablic: asteroidów oraz pocisków

***function ZderzenieStatkuzAsteroida()***

***{***

// inicjuj nową pustą tablicę dla obiektów klasy Asteroida

***asteroidy = new Array();***

// inicjuj nową pustą tablicę obiektów / pocisków statku kosmicznego

***pociski = new Array();***

***}***

/\* >>>>>>>>>>>>>>>>>>>>>>>>>>> ASTEROIDY - KONIEC <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

### Zmiany w innych miejscach kodu

W celu uaktywnienia rysowania asteroidów należy w funkcji ***GlownaGra()*** (wewnątrz instrukcji ***if (!gra.koniec)*** za komentarzem:

// wywołanie funkcji rysowania asteroid umieść wiersz

***RysujAsteroidy();***

Oprócz tego wróć do sekcji programu głównego w którym

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PROGRAM GŁÓWNY GRY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

za komentarzem:

// Twórz nową asteroidę co 1000 ms umieść wiersz

***setInterval(UtworzAsteroide, 1000);***

Użycie metody ***setInterval*** daje nam możliwość tworzenia nowych asteroid niezależnie od przebiegu głównej animacji.

Pozostało nam jeszcze uzupełnić funkcję ***DetekcjaKolizjiPociskowZAsterodiami*** z sekcji **Pociski,** która posiada

komentarz // kod będzie uzupełniony później na etapie 4g – asteroidy.

Detekcja kolizji pocisku z asteroidą powoduje oznaczenie asteroidy, że została zniszczona oraz to, że pocisk trafił cel. Ponieważ trafienie w asteroidę jest punktowane, więc zwiększymy też wartość punktów w grze.

// Funkcja dokonująca detekcji kolizji pocisków z asteroidami

***function DetekcjaKolizjiPociskowZAsterodiami(i)***

***{***

// Sprawdź, czy pocisk trafił w asteroidę

// detekcja kolizji pocisków z asteroidami

***for (var a = 0; a < asteroidy.length; a++) {***

***if ((Math.abs(asteroidy[a].X - pociski[i].X) < 35) && (Math.abs(asteroidy[a].Y - pociski[i].Y) < 32)) {***

// odtwórz dźwięk

***asteroidy[a].DzwiekWybuchuAsteroidy.play();***

// oznacz asteroidę do usunięcia

***asteroidy[a].CzyZniszczona = true;***

// pocisk, który trafił w cel oznaczony jest przez wartość true dla CzyTrafilCel

***pociski[i].CzyTrafilCel = true;***

// zwiększ punkty jeśli nastąpiło uderzenie pocisku w asteroidę

***gra.punkty++;***

***}***

***}***

***}***

Otwórz plik index.html w przeglądarce Chrome i sprawdź działanie pojawiających się losowo asteroid.

Czy udaje się je zestrzelić?

## 4h) Tablica wyników - liczba punktów, czas gry, liczba żyć,

Teraz zajmiemy się wyświetleniem rezultatów w obszarze kanwy rezultaty. Do tego celu będzie nam potrzebna funkcja ***RysujRezultaty()***. Zadaniem tej funkcji będzie: rysowanie/wyświetlanie liczby zdobytych punktów, liczby żyć oraz czasu gry, który w następnym rozdziale zostanie zaktualizowany do odpowiedniego formatu cyfrowego.

W sekcji /\* >> RYSUJ REZULTATY – POCZĄTEK/KONIEC << \*/ umieść poniższy kod

/\* >>>>>>>>>>>>>>>>> RYSUJ REZULTATY - POCZĄTEK <<<<<<<<<<<<<<<<<<<<< \*/

/\* \*\*\*\*\*\*\*\* Funkcja rysująca wyniki na canvasie rezultaty \*\*\*\*\*\*\*\* \*/

***function RysujRezultaty()***

***{***

***rezctx.fillStyle = "rgb(0,100,200)";***

***rezctx.fillRect(0, 0, rezultaty.width, 100);***

***rezctx.fillStyle = 'rgb(0,0,0)';***

***rezctx.font = "bold 2.0em sans-serif";***

***rezctx.fillText("Punkty: " + gra.punkty, 20, rezultaty.height/2+10);***

***rezctx.fillText("Czas: " + gra.czaszapiscyfrowy, rezultaty.width/2 - 150, rezultaty.height/2+10);***

***rezctx.fillText("Życia: " + gra.zycia, rezultaty.width - 180, rezultaty.height/2+10);***

//jeśli nastąpiło zderzenie z asteroidą to rysujemy dodatkowy tekst na canvasie rezultaty

***if (gra.zderzenie > 0)***

***{***

// kolor czerwony wypełnienia

***rezctx.fillStyle = 'rgb(255,0,0)';***

***rezctx.font = "bold 1.2em sans-serif";***

***rezctx.fillText("Liczba zderzeń z asteroidą: " + gra.zderzenie, rezultaty.width/2 - 150, rezultaty.height-8);***

***}***

***}***

/\* >>>>>>>>>>>>>>>>> RYSUJ REZULTATY - KONIEC <<<<<<<<<<<<<<<<<<<<< \*/

### Zmiany w innych miejscach kodu

W celu uaktywnienia rysowania rezultatów należy w funkcji ***GlownaGra()*** za komentarzem:

// wywołanie funkcji rysującej rezultaty umieść wiersz

***RysujRezultaty();***

Otwórz plik index.html w przeglądarce Chrome i sprawdź czy pojawiają się na kanwie rezultaty wyniki gry.

Na chwilę obecną czas rozgrywki nie jest jeszcze wyświetlany. Wyświetlanie czasu wykonamy w kolejnym rozdziale.

## 4i) Czas gry

Ważną informacją dla gracza jest czas jaki spędza podczas grania. Z tego powodu potrzebna nam będzie klasa obiektowa ***CzasGry*** ze składowymi związanymi z elementami czasu. Przyda się też funkcja ***CzasTrwaniaGry*** przeliczająca czas na format ***HH:MM:SS***, której zadaniem będzie także aktualizacja składowej ***czaszapiscyfrowy*** zmiennej obiektowej ***gra***. Aby tykający czas był niezależny od animacji lepiej/wygodniej wywołać funkcję ***CzasTrwaniaGry*** za pomocą metody ***setInterval co 1000 ms***.

W sekcji /\* >> CZAS GRY – POCZĄTEK/KONIEC <<< \*/ umieść poniższy kod

/\* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> CZAS GRY - POCZĄTEK <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

// konstruktor klasy CzasGry

***function CzasGry(H,M,S)***

***{***

***this.godzina = H;***

***this.minuta = M;***

***this.sekunda = S;***

***}***

// utworzenie nowej zmiennej obiektowej typu CzasGry

***var czas = new CzasGry(0,0,0);***

/\* >>>>>>>>>>>>>>>>>>>>>>>>>>>> KONIEC CzasGry <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< \*/

// Funkcja przeliczająca czas trwania gry

// zapis czasu w formacie HH:MM:SS

***function CzasTrwaniaGry()***

***{***

***var HH, MM, SS;***

***czas.sekunda++;***

***if (czas.sekunda == 60 )***

***{***

***czas.minuta++;***

***czas.sekunda = 0;***

***}***

***if (czas.minuta == 60)***

***{***

***czas.godzina++;***

***czas.minuta = 0;***

***czas.sekunda = 0;***

***}***

***if (czas.godzina == 24)***

***{***

***czas.godzina = 0;***

***czas.minuta = 0;***

***czas.sekunda = 0;***

***}***

***if (czas.godzina < 10 ) HH = "0" + czas.godzina***

***else HH = czas.godzina;***

***if (czas.minuta < 10 ) MM = "0" + czas.minuta***

***else MM = czas.minuta;***

***if (czas.sekunda < 10 ) SS = "0" + czas.sekunda***

***else SS = czas.sekunda;***

***// uaktualnienie składowej czasu zmiennej gra***

***gra.czaszapiscyfrowy = HH + ":" + MM + ":" + SS ;***

***}***

/\* >>>>>>>>>>>>>>>>>>>>>>>> CZAS GRY - KONIEC <<<<<<<<<<<<<<<<<<<<<<<<<< \*/

### Zmiany w innych miejscach kodu

W celu uaktywnienia przeliczania czasu umieść ***funkcję CzasTrwaniaGry*** w sekcji programu głównego za komentarzem:

// uruchom timer co 1 sekundę z przeliczeniem czasu gry na zapis cyfrowy-> w następujący sposób

***setInterval(CzasTrwaniaGry, 1000);***

Otwórz plik index.html w przeglądarce Chrome i sprawdź czy podczas rozgrywki pojawia się na kanwie rezultaty odpowiedni czas w formacie HH:MM:SS.

## 4j) Koniec gry – „game over”

Po straceniu wszystkich żyć powinien zostać zaprezentowany komunikat, który poinformuje uczestnika o fakcie, że gra zakończyła się oraz co zrobić, aby rozpocząć grę od nowa. Wszystkie te czynności wykona dla nas funkcja ***KoniecGry()***.

W sekcji /\* >> INNE - FUNKCJE – POCZĄTEK/KONIEC << \*/ za komentarzem

/\* \\\\\\\\\\\\\\ Funkcja KoniecGry //////////////// \*/ umieść poniższy kod

***function KoniecGry()***

***{***

// kolor biały wypełnienia

***glowctx.fillStyle = 'rgb(255,255,255)';***

// wybór czcionki, rozmiaru, stylu

***glowctx.font = "bold 2.8em sans-serif";***

// narysowanie tekstu koniec gry

***glowctx.fillText("Koniec gry!", GlownyCanvas.width/2-140, GlownyCanvas.height/2 - 20);***

// kolor czerwony wypełnienia

***glowctx.fillStyle = 'rgb(255,0,0)';***

// wybór czcionki, rozmiaru, stylu

***glowctx.font = "bold 1.8em sans-serif";***

// narysowanie tekstu F5 zagrania ponownie

***glowctx.fillText("Klawisz F5 - zagraj ponownie!", GlownyCanvas.width/2-200, GlownyCanvas.height/2+20);***

***}***

### Zmiany w innych miejscach kodu

W celu uaktywnienia działania funkcji ***KoniecGry*** należy w funkcji ***GlownaGra()*** za komentarzem:

// wywołaj funkcję z informacją, że nastąpił koniec gry umieść wiersz

***KoniecGry();***

Otwórz plik index.html w przeglądarce Chrome i rozegraj rozgrywkę starając się utracić 3 życia. Sprawdź, czy pojawił się komunikat, że gra została zakończona oraz to że można rozpocząć grę odnowa.

# Pytania dotyczące gry. Wady i zalety gry.

Pytania:

Co należy zrobić, aby zwiększyć prędkość spadania asteroid, a co żeby częstotliwość pojawiania się w grze asteroid zwiększyła się?

Wady i zalety: Wszystkie programy, a tym samym i gry mają swoje wady i zalety. Zastanów się, czy po zakończeniu rozgrywki nasza gra webowa przestała działać i nie wykonuje już żadnych czynności? Swoją odpowiedź uzasadnij. Jeśli potrafisz to dopisz fragmenty kodu, które według Ciebie powinny zostać uwzględnione na etapie projektowania. Co należałoby w tej grze udoskonalić lub zmienić w przyszłości?