Implementacja

W tej części kursu utworzysz nową stronę, dodasz do niej elementy graficzne i wyświetlisz je odpowiednio w elemencie Canvas za pomocą JavaScript. Wynik zadań został przedstawiony na Rys. 1.

Rys. 1. Świat gry.

Przygotowanie strony

Pierwszym krokiem będzie przygotowanie w programie strony internetowej.

4. Umieść niezbędne elementy do obsługi elementu Canvas:

- w znaczniku title wpisz:

```
Alien Game w HTML5
```

- w elemencie body umieść znacznik cavas:

```
<canvas id="gra" width="480" height="800"></canvas>
```

Programowanie elementu Canvas

W tym momencie mamy stronę internetową z elementem Canvas. Teraz wszystkie operacje będziemy wykonywali w języku JavaScript.

5. W JavaScript sprawdź, czy przeglądarka wspiera element Canvas:

- ustaw kursor przed znacznikiem </body> , wstaw nową linię i dodaj następujący kod:

```
<script type="text/javascript">
  //po uruchomieniu strony wykonaj:
  window.addEventListener("load", function () {
      // Sprawdź, czy przeglądarka obsługuje element Canvas
      var canvas = document.getElementById('gra');
          if (!canvas.getContext) { return; }
      ctxMain= canvas.getContext('2d');
      width = canvas.width;
      height = canvas.height;

      // Dodaj bufor dla elementu canvas
      canvasBuffer = document.createElement('canvas');
      canvasBuffer.width = width;
      canvasBuffer.height = height;
      ctx = canvasBuffer.getContext('2d');

      // Pobierz zasoby
      LoadResources();
  }, false);
// Funkcja pobierająca zasoby
// Funkcja rysująca świat gry
// Funkcje do sterowania czołgiem
// Zmienne w aplikacji
</script>
```

Informacja
W celu uzyskania płynnego ruchu na elemencie canvas utworzyliśmy dodatkowy element canvas – bufor. Wszystkie operacje rysowania będziemy wykonywali na nim, a po wyrysowaniu, jednorazowo, całą jego zawartość przerysujemy na widocznym elemencie canvas.

6. Dodaj zmienne do wyświetlenia świata gry:

- ustaw kursor po // Zmienne, w aplikacji wstaw nową linię i dodaj następujący kod:

```
var ctxMain=null;
var canvasBuffer;
var ctx = null;
var width = 0;
var height = 0;
//liczba elementów do pobrania
var doPobrania = 0;
//elementy graficzne
var background, sky, mountains, hills, ground, tank, cloud1, cloud2, sun, bullet, tire, alien;
//pozycja słońca
var sunPos = 0;
//pozycja X czołgu
var tankXTarget;
```

Pobierz zasoby gry

Teraz, gdy mamy już zdefiniowane zmienne, możemy pobrać zasoby. Zasoby w grach, to przede wszystkim elementy graficzne, symbolizujące obiekty w grze (np. gracza, pociski, itp.) oraz muzykę odtwarzaną w tle.

7. Dodaj funkcję pobierającą niezbędne grafiki:

- ustaw kursor po // Funkcja pobierająca zasoby. Wstaw nową linię i dodaj następujący kod:

```
function LoadResources() {
    mountains = loadImg('Images/mountains_blurred.png');
    hills = loadImg('Images/hills.png');
    ground = loadImg('Images/ground.png');
    tank = loadImg('Images/tank.png');
    cloud1 = loadImg('Images/cloud1.png');
    cloud2 = loadImg('Images/cloud2.png');
    sun = loadImg('Images/sun.png');
    bullet = loadImg('Images/bullet.png');
    alien = loadImg('Images/alien1.png');
    setTimeout(checkLoad, 100);
}
function loadImg(path) {
    doPobrania++;
    var v = new Image();
    v.src = path;
    v.addEventListener("load", function () { doPobrania--; }, false);
    return v;
}
function checkLoad() {
    if (doPobrania == 0) {
        DrawGameWorld();
    }
    else {
        setTimeout(checkLoad, 100);
    }
}
```

Informacja
Funkcja **loadImg()** umożliwia pobranie obrazka umieszczonego w określonej lokalizacji. Zmienna **doPobrania** określa liczbę obrazków pozostałą do pobrania (zwiększana przy pobraniu, zmniejszana po zdarzeniu load obrazka).

Funkcja **checkLoad()**, uruchamiana po 100ms od ostatniego wywołania funkcji **loadImg()**, sprawdza, czy wartość zmiennej **doPobrania** jest równa 0. Jeśli tak jest, to uruchamiana jest funkcja rysująca świat gry, jeśli nie, to po kolejnych 100ms uruchamiana jest ponownie funkcja **checkLoad()**;

Wyrysuj świat gry

Świat gry jest najczęściej światem fikcyjnym, w którym ma miejsce akcja gry.

8. Wyrysuj świat gry.
9. Ustaw kursor po // Funkcja rysująca świat gry i dodaj następujący kod:

```
function DrawGameWorld() {
  ctx.clearRect(0, 0, width, height);
  // Niebo
  ctx.fillStyle = '#6495ED';
  ctx.fillRect(0, 0, width, height);
  // Góry
  ctx.drawImage(mountains, 0, height - 401);
  // Pagórki
  ctx.drawImage(hills, 0, height - 175);
  // Ziemia
  ctx.drawImage(ground, 0, height - 86);
  // Słońce
  sunPos = sunPos + .1;
  ctx.drawImage(sun, sunPos, 0);
  // Chmurka 1
  ctx.drawImage(cloud1, 150, 50);
  // Chmurka 2
  ctx.drawImage(cloud2, 300, 20);
  // Czołg
  ctx.drawImage(tank, 300, 644);
  // Obcy
  ctx.drawImage(alien, 100, 200);
  // Pociski
  // Wyniki
  // Prześlij bufor na element canvas
  ctxMain.drawImage(canvasBuffer, 0, 0);
}
```

Zapisz zmiany. Wciśnij przycisk Run. Sprawdź, czy na stronie możesz zobaczyć widok podobny do przedstawionego na Rys. 1.

Sterowanie za pomocą klawiatury

Sterowanie za pomocą klawiatury polega na nasłuchiwaniu zdarzenia wciśnięcia klawisza (keydown) i określeniu funkcji odpowiedzialnej za jego obsługę przy pomocy:

document.addEventListener("keydown", KeyDown, false);

W funkcji **KeyDown** sprawdzamy, który klawisz został wciśnięty i na tej podstawie zmieniamy elementy gry. Kody wybranych klawiszy zostały przedstawione w poniższej tabeli:

Klawisz	Kod
Spacja	32
Strzałka lewo	37
Strzałka góra	38
Strzałka prawo	39
Strzałka dół	40

1. Dodaj nasłuchiwanie zdarzenia wciśnięcia klawisza na klawiaturze:

- poniżej linijki LoadResources(); dodaj:

```
// Nasłuchiwanie  klawiatury
document.addEventListener("keydown", KeyDown, false);
// Nasłuchiwanie  myszki
```

2. Dodaj funkcje odpowiedzialne za zmianę pozycji czołgu:

- poniżej linijki // Funkcje do sterowania czołgiem dodaj:

```
function KeyDown(e) {
    // Klawisz strzałka w lewo
    if (e.keyCode == 37) {
      tankXTarget -= 10;
       dir = -1;
    } else
      // Klawisz strzałka w lewo
      if (e.keyCode == 39) {
        tankXTarget += 10;
        dir = 1;
      } else
        // Klawisz spacja
        if (e.keyCode == 32)
          Fire();
}
// funkcja odpowiedzialna za strzały
function Fire() { }
```

Informacja
Funkcja KeyDown wywołana jest w momencie wciśnięcia przez użytkownika klawisza na klawiaturze. Jeśli będzie to klawisz strzałki prawej lub lewej, to zostanie zmieniona wartość zmiennej tankXTarget, odpowiedzialnej za pozycję czołgu w grze. Jeżeli będzie to przycisk spacji, to zostanie wywołana funkcja Fire.

Funkcja Fire odpowiedzialna jest za strzelanie pociskami.

Sterowanie za pomocą myszki

Sterowanie za pomocą myszki polega na nasłuchiwaniu dwóch zdarzeń - ruchu myszki (mousemove) oraz wciśnięcia lewego klawisza myszki (mousedown), i określeniu funkcji odpowiedzialnych za ich obsługę.

1. Dodaj nasłuchiwanie zdarzeń związanych z obsługą myszki:

- poniżej linijki // Nasłuchiwanie myszki dodaj:

```
document.addEventListener("mousemove", Follow, false);
document.addEventListener("mousedown", Fire, false);
```

2. Dodaj funkcje odpowiedzialne za zmianę pozycji czołgu:

- poniżej linijki // Funkcje do sterowania czołgiem dodaj:

```
function Follow(e) {
      tankXTarget = e.clientX;
  }
```

3. Wyświetl czołg na pozycji określonej przez zmienną tankXTarget:

- w funkcji DrawGameWorld zastąp linijkę: ctx.drawImage(tank, 300, 644); linijką:

```
ctx.drawImage(tank, tankXTarget - 43, 644);
```

4. Zapisz zmiany. Wciśnij przycisk **Run**. Sprawdź, czy możesz sterować czołgiem za pomocą klawiatury (strzałka prawo/lewo i myszki).

Informacja
W tym momencie nie możemy sterować czołgiem, ponieważ rysowanie świata gry odbywa się tylko jeden raz. W celu poruszenia czołgiem musimy cyklicznie, co określoną liczbę klatek na sekundę, przerysowywać element Canvas.

Cykliczne przerysowywanie elementu Canvas

W celu cyklicznego przerysowywania musimy, co określony czas, wywoływać funkcję odpowiedzialną za wyrysowywanie świata gry. Do cyklicznego wywołania funkcji służy polecenie:

```
setInterval(a,b)
```

gdzie a jest nazwą wywoływanej funkcji, b jest czasem w milisekundach. W grach posługujemy się pojęciem klatki na sekundę (FPS). W celu przeliczenia czasu na FPS wykorzystujemy następujący wzór:

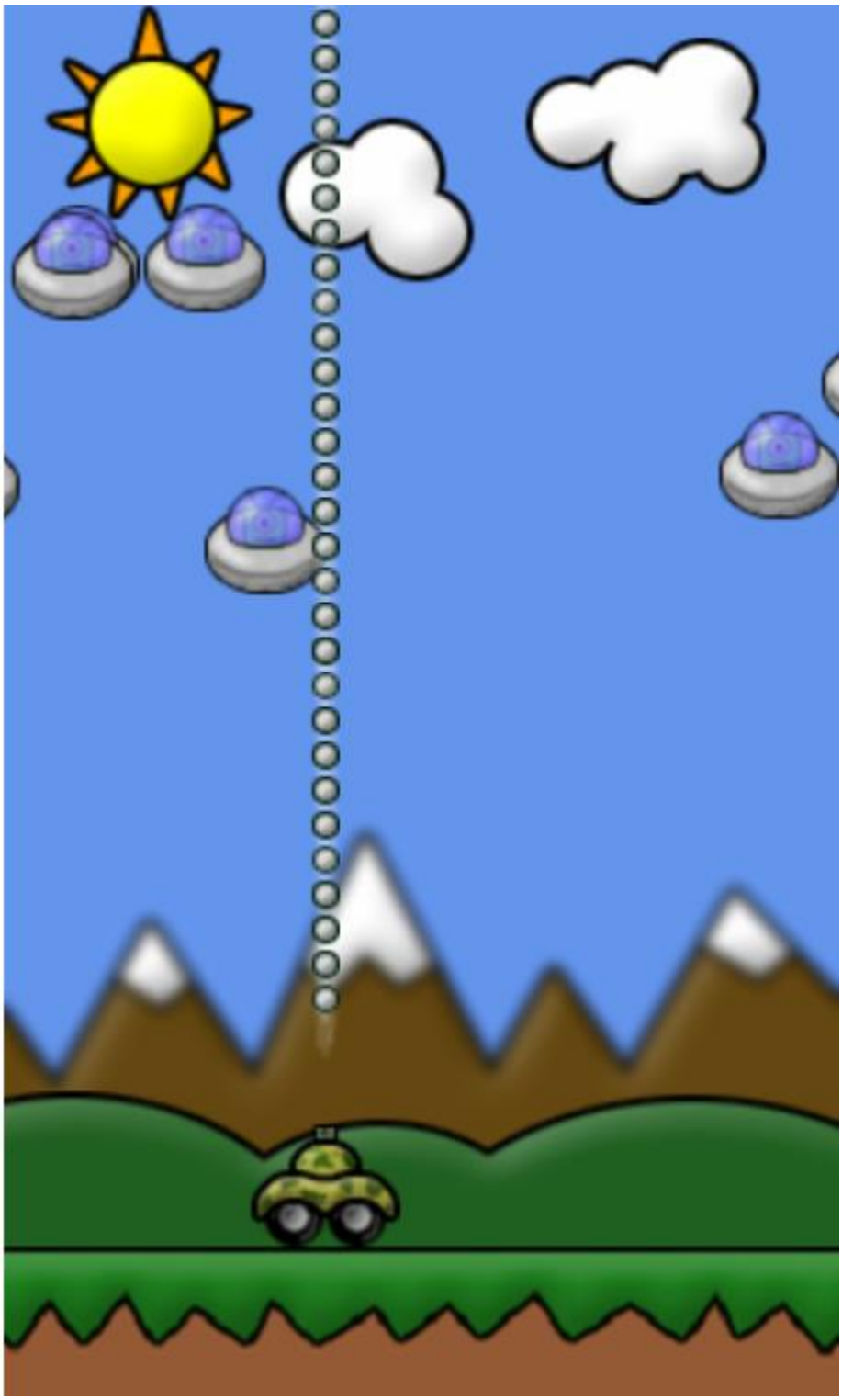
$$czas[ms] = \frac{1000}{FPS}$$

1. Wywołaj funkcję DrawGameWorld, z częstotliwością 16 klatek na sekundę:

- w funkcji checkLoad zamień linijkę DrawGameWorld();na:

```
setInterval(DrawGameWorld, 1000 / 16);
```

2. Zapisz zmiany. Wciśnij przycisk Run. Sprawdź, czy możesz sterować czołgiem za pomocą klawiatury (strzałka prawo/lewo i myszki).
3. W tej części kursu wykorzystamy stronę, którą przygotowaliśmy w poprzednim odcinku. Dodamy do niej możliwość poruszania dla przeciwników – obcych oraz strzelania przez nasz czołg. Wynik zadania został przedstawiony na Rys. 1.

4. 
5. ***Rys. 1. Obcy i pociski w ruchu.***
6. Przechowując informacje o wielu elementach, których liczby nie znamy – nie wiemy ilu obcych będzie jednocześnie na ekranie oraz ile pocisków wystrzeli użytkownik. Do przechowywania elementów jednego typu, o zmiennej liczbie elementów, służy struktura zwana tablicą dynamiczną - **Array**. Tworzymy ją za pomocą:

```
var bullets = new Array();
```

Pomocne metody i właściwości tablicy:

- .length – podaje rozmiar tablicy,
- .push – dodaje element do tablicy na końcu,
- .splice(a,b) – wycina z tablicy b elementów, począwszy od elementu a.

Pociski na stronie

Przechowując dane o pocisku, potrzebujemy zapisać więcej niż jedną informację. Przechowywanie informacji w wielu tablicach (np. tablica pozX, pozY,…) jest niepraktyczne. Najlepiej przygotować strukturę, która będzie przechowywała wszystkie potrzebne informacje. W naszym przykładzie utworzymy obiekt Bullet:

```
var Bullet = function (x, y) {
            this.X = x;
            this.Y = y;
            this.IsDirty = false; }
```

zawiera on następujące informacje:

- X – pozycja X pocisku,
- Y – pozycja Y pocisku,
- IsDirty – zmienna informująca o konieczności usunięcia pocisku po trafieniu we wrogi obiekt.

1. Przygotuj zmienne do przechowywania informacji o pociskach:

- na końcu kodu JavaScript, po linii var tankXTarget; wklej:

```
// Zmienne przechowujące informacje o pociskach
  var bullets = new Array();
  var Bullet = function (x, y) {
      this.X = x;
      this.Y = y;
      this.IsDirty = false; }
// Zmienne przechowujące informacje o obcych
```

2. W funkcji Fire() dodaj kolejny pocisk na pozycji, w której znajduje się czołg:

- zamień linijkę function Fire() { } na:

```
function Fire() {
  bullets.push(new Bullet(tankXTarget - 8, 644)); }
// Funkcja wyświetlająca pociski
```

3. Dodaj funkcję odpowiedzialną za wyświetlenie wszystkich pocisków z tablicy:

- poniżej poprzednio dodanego kodu dopisz:

```
function DrawBullets() {
  // Usuń pociski z listy
  for (var i = 0; i < bullets.length; i++) {
      if (bullets[i].IsDirty == true || bullets[i].Y <= 0) {
          bullets.splice(i, 1);
          i--;}
  }
  // Narysuj pociski
  var hit = false;
  for (var i = 0; i < bullets.length; i++) {
      bullets[i].Y -= 20;
      // Sprawdź, czy pocisk trafił któregoś obcego
      if (!hit)
          ctx.drawImage(bullet, bullets[i].X, bullets[i].Y);
  }
}

// Funkcja wyświetlająca obcych
```

Wyświetl pociski na stronie:

· w funkcji DrawGameWorld(), poniżej linijki // Pociski dopisz:

```
DrawBullets();
```

5. Zapisz zmiany. Wciśnij przycisk Run. Sprawdź, czy możesz strzelać za pomocą lewego przycisku myszki i spacji.

Obcy na ekranie

Teraz musimy przygotować informacje o obcych, którzy będą pojawiali się na ekranie. Obcy, oprócz informacji, które wcześniej określiliśmy dla pocisków, muszą dodatkowo mieć informację o prędkości poruszania się po ekranie. Nowy obiekt pojawia się na ekranie co 1 sekundę (1000 milisekund), na pozycji x równej - 70px (poza ekranem), na pozycji y - w przedziale 100-200 px i z prędkością 0 -10.

1. Przygotuj zmienne do przechowywania informacji o obcych:

- po linii // Zmienne przechowujące informacje o obcych dodaj:

```
var aliens = new Array();
var Alien = function (x, y, s) {
    this.X = x;
    this.Y = y;
    this.Speed = s;
    this.IsDirty = false; }
```

Informacja
Obcy posiada dodatkowy parametr, zawierający informację o prędkości poruszania się po świecie gry.

2. Dodaj funkcję wyświetlającą obcych:

- po linii // Funkcja wyświetlająca obcych dodaj:

```
function DrawAliens() {
  // Usuń obcego  z ekranu
  for (var i = 0; i < aliens.length; i++) {
      if (aliens[i].IsDirty == true || aliens[i].X > width) {
          aliens.splice(i, 1);
          i--; }
  }
  // Narysuj obcego
  for (var i = 0; i < aliens.length; i++) {
      aliens[i].X += aliens[i].Speed;
      ctx.drawImage(alien, aliens[i].X, aliens[i].Y);
  }
}
// Funkcja dodająca obcych do listy
```

3. Wyświetl obcych na ekranie:

- w funkcji DrawGameWorld zmień linijkę ctx.drawImage(alien, 100, 200); na:

```
DrawAliens();
```

Informacja
Pozostał nam jeden problem, jak dodawać obcych do gry. Najprościej – cyklicznie, co 1s dodać nowego obcego.

4. Dodaj funkcję dodawania obcego do listy:

- poniżej linijki // Funkcja dodająca obcych do listy wpisz:

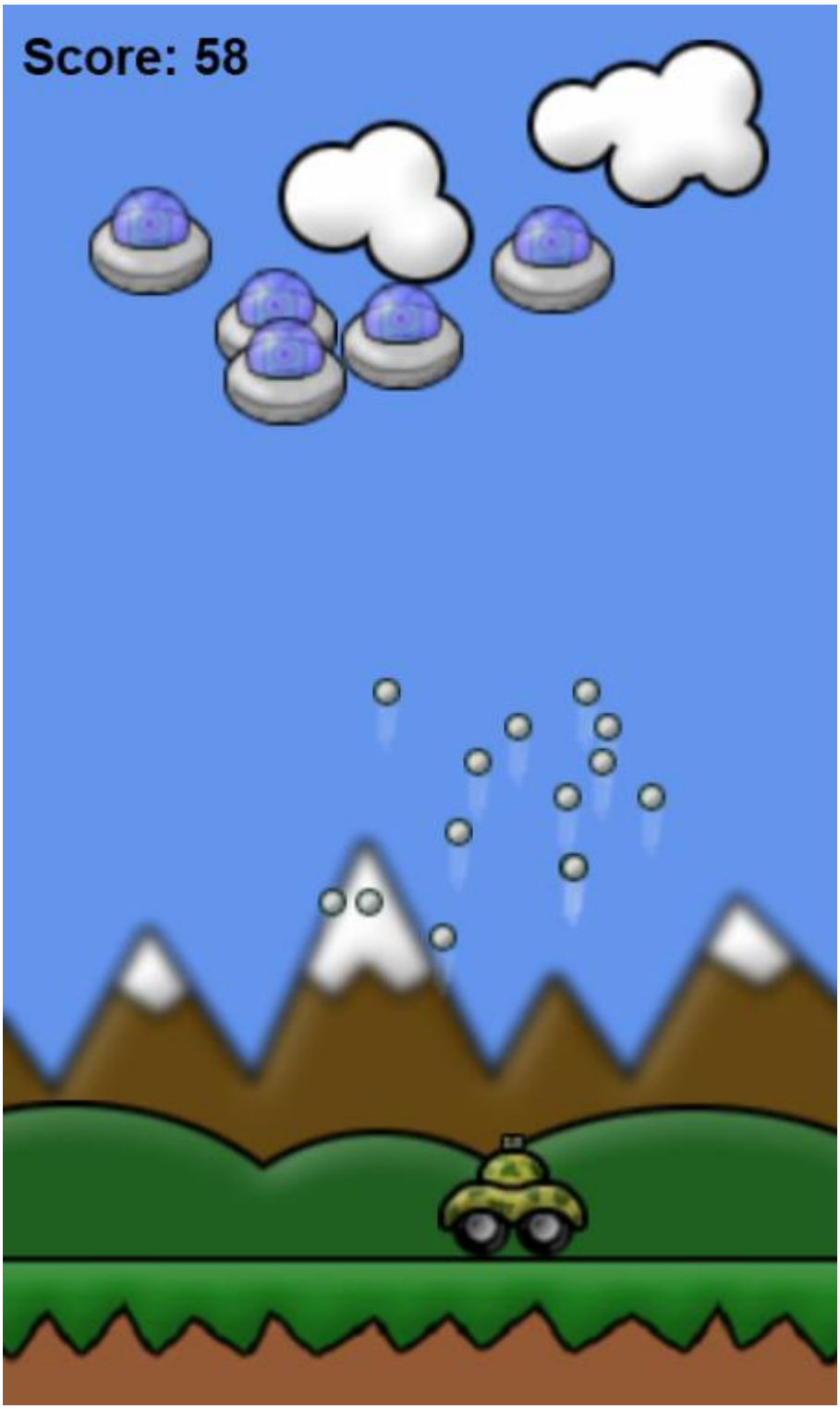
```
function CreateAlien() {
   aliens.push(new Alien(-70, 100 + Math.random() * 200, Math.random() *
10)); }
```

5. Dodaj cykliczne wywołanie funkcji CreateAlien:

- w funkcji checkLoad, za linijką setInterval(DrawGameWorld, 1000 / 16); dodaj:

```
setInterval(CreateAlien, 1000);
```

6. Zapisz zmiany. Wciśnij przycisk Run. Sprawdź, czy pojawiają się nowe statki na ekranie.
7. W tej części kursu wykorzystamy stronę, którą przygotowaliśmy w poprzednim odcinku kursu. Dodamy do niej wykrywanie kolizji oraz wyświetlanie informacji o zdobytych punktach. Wynik zadania został przedstawiony na Rys. 1.

8. 

9. ***Rys. 1. Obcy i pociski w ruchu.***
10. Wykrywanie kolizji to sprawdzenie, czy przemieszczenie obiektu w danym kierunku jest możliwe, czy nie znajdują się na jego drodze jakieś przeszkody, a jeżeli tak, to należy podjąć decyzję, w przypadku gdy coś dzieje się z obiektem i przeszkodą. W naszym przypadku skupimy się na sprawdzeniu, czy wystrzelony pocisk nie znajduje się w obszarze obcego, jeśli tak, to usuniemy pocisk i obcego z ekranu gry, dodamy punkty za trafienie i wyświetlimy na ekranie wynik.

Pociski na stronie

Pocisk porusza się pionowo od miejsca, w którym wystrzelił go użytkownik. Jego pozycja przechowywana jest w obiekcie Bullet w tablicy bullets. Obcy porusza się ze strony lewej na prawą. Jego pozycja przechowywana jest w obiekcie Alien w tablicy aliens.

Na Rys. 2. Przedstawiona została sytuacja, w której pocisk zbliża się do obcego. Obcy i pocisk zostały obrysowane czarnymi prostokątami, aby przedstawić obszar działania kolizji. Pozycja (0,0) dla obiektów znajduje się w lewym górnym rogu, zatem, jeżeli odległość między pozycjami wynosi: X < 70 oraz Y <62 (70x60 – wielkość obrazka obcego), to oba obiekty, po trafieniu, usuwane są z gry i dodany zostaje punkt dla użytkownika.

Rys. 2. Kolizje obiektów w grze.

Niestety, kolizję wykrywamy na przecięciu dwóch prostokątów – obcego i pocisku, co powoduje, że jeśli „trafimy" w prawą lub lewą stronę, to usuniemy obcego, mimo że pocisk nie doszedł do krawędzi obcego. W przypadku obiektów bardziej skomplikowanych należy wziąć pod uwagę kilka obszarów prostokątnych.

1. Sprawdź, czy pocisk znajduje się obok obcego:

- w funkcji DrawBullets(), po linijce // sprawdź, czy pocisk trafił któregoś obcego wpisz:

```
for (var a = 0; a < aliens.length && hit == false; a++) {
    if ((Math.abs(aliens[a].X - bullets[i].X) < 70) && (Math.abs(aliens[a].Y - bullets[i].Y) < 62)) {
      aliens[a].IsDirty = true;
      bullets[i].IsDirty = true;
      hit = true;
    }
}
```

2. Zapisz zmiany. Wciśnij przycisk **Run**. Sprawdź, czy możesz trafić pociskiem statki na ekranie.

Dodawanie punktów

Teraz musimy zliczać i wyświetlać punkty za trafienia.

1. Dodaj zmienną przechowującą zliczane punkty:

- w linijce powyżej </script> dodaj:

```
// Zmienna przechowująca punkty
var score = 0;
```

2. Dodaj funkcję wyświetlającą liczbę punktów:

- powyżej linijki // Funkcja wyświetlająca obcych dodaj:

```
function DrawScore() {
  ctxMain.fillStyle = 'rgb(0,0,0)';
  ctxMain.font = "bold 1.8em sans-serif";
  ctxMain.fillText("Score: " + score, 12, 40);
}
```

3. Dodaj wywołanie funkcji wyświetlającej w funkcji DrawGameWorld():

- w funkcji DrawGameWorld(), poniżej linijki ctxMain.drawImage(canvasBuffer, 0, 0); dopisz:

```
DrawScore();
```

4. Zwiększ zmienną score przy każdym wykryciu kolizji:

- w funkcji DrawBullets(), po linijce hit = true; dodaj:

```
score++;
```

5. Zapisz zmiany. Wciśnij przycisk **Run**. Sprawdź, czy punkty są poprawnie liczone i wyświetlane.

Informacja
Zauważ, że punkty musimy wypisać bezpośrednio na głównym elemencie Canvas, a nie na buforze, ponieważ tekst nie może zostać przerysowany przez metodę drawImage.