Gra zrealizowana

w JavaScript

-

Space Shootout

Autor:

Łukasz Kurdziel

Spis treści

1.Wstęp **3**

2.Wykorzystane Technologie **3**

3.1.HTML 3

3.2.Java Script 3

3.3.Canvas 3

3.4.CSS. 3

3.Funkcjonalność aplikacji **4**

3.1.Początek/Koniec Gry 5

3.2.Zasada działania gry 7

3.3.Obiekt gracza 8

3.3.1.Rysowanie gracza 10

3.4.Antagoniści 12

3.4.1.Pojawianie się przeciwników 13

3.5.Walka 15

4.Podsumowanie **18**

5.Literatura **18**

1.Wstęp

Celem pracy jest stworzenie gry na bazie języka JavaScript oraz przy użyciu elementu Canvas. Space Shootout to gra, gdzie gracz wciela się w kapitana statku kosmicznego i stara się przeżyć nieskończoną inwazje wrogich statków. Gra trwa dopóki gracz nie utraci wszystkich punktów zdrowia. Gracz może manewrować pomiędzy nacierającymi jednostkami wroga oraz strzelać w nie pociskami.

2.Wykorzystane technologie

W grze zostały użyte następujące technologie.

2.1.HTML

Kod, który służy do tworzenia struktury strony, jej zawartości, a także łączenia poszczególnych elementów między sobą. Struktury stron są budowane na zasadzie łączenia kontenerów takimi jak akapity, hiperłącza, formularze itp.

2.2.JavaScript

JavaScript to skryptowy język programowania, który znany jest z umożliwienia wdrożenia bardziej zaawansowanych elementów na stronach internetowych.

2.3.Canvas

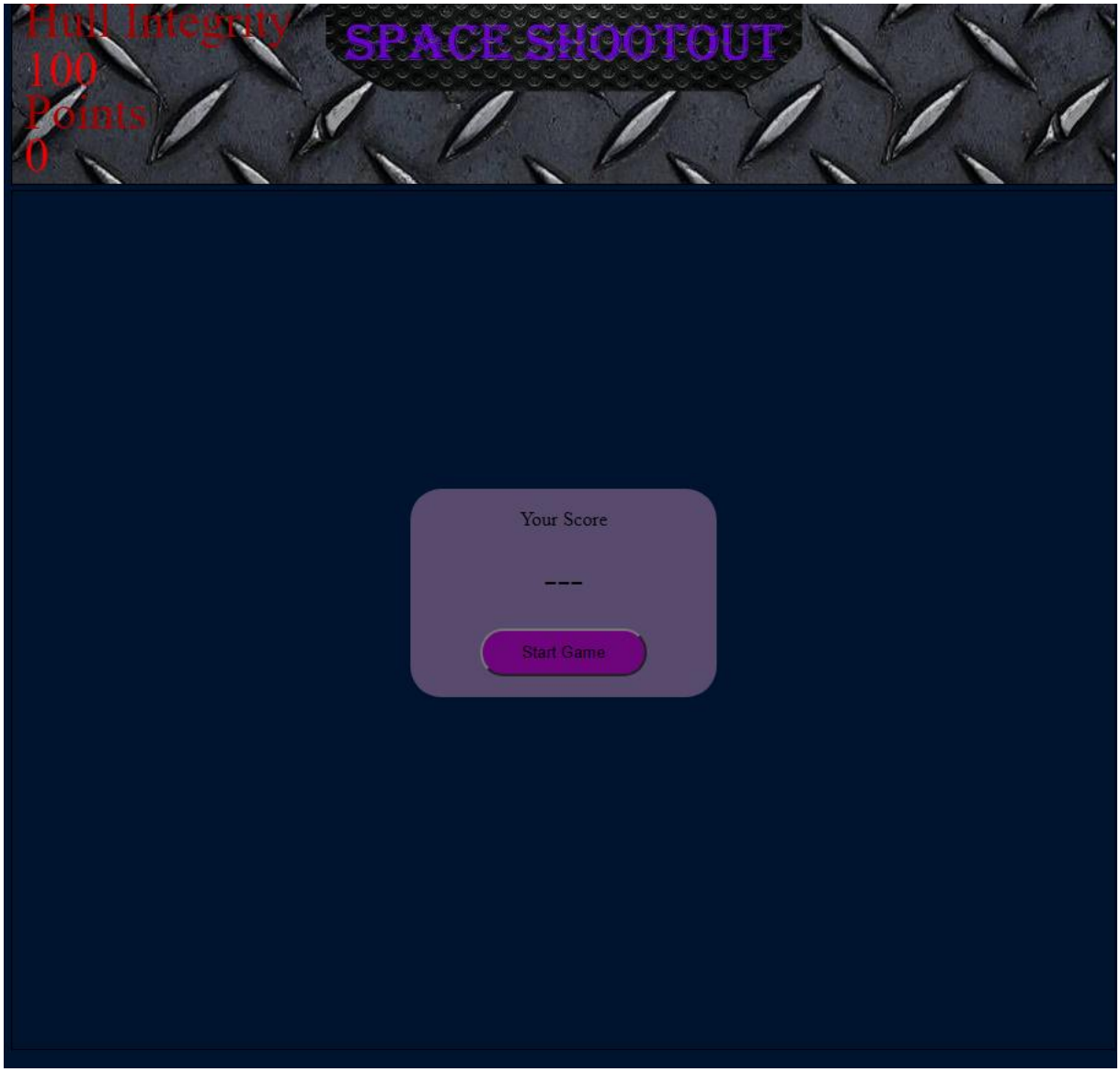
Jest to element języka HTML, który pozwala na rysowanie grafik używając skryptów.

3.4.CSS

Kaskadowe arkusze stylów umożliwiają definiowanie stylu i układu graficznego na dokumentach HTML

3.Funkcjonalność aplikacji

Podstawowy widok gry został stworzony w HTML oraz dwóch obiektów Canvas.



Rys.4.0 Widok w HTML

Pierwszy obiekt Canvas działa jako podstawowy interfejs użytkownika, na którym są wyświetlone pozostałe punkty zdrowia gracza oraz zdobyte punkty w obecnej grze.

Drugi obiekt Canvas, na którym jest wyświetlana właściwa gra jest początkowo zasłonięty przez okno dialogowe czekające na rozpoczęcie gry przez gracza.

3.1 Początek/Koniec gry.

To okno dialogowe odpowiada za rozpoczęcie i restartowanie gry. Jest ono częścią widoku HTML

```
<div class="gameOverMenu" id="gameOverEl">
    <center>

        <p>Your Score</p>
        <h1 id="scoreEl">---</h1>
        <div>
        <button class="restart" id="restartGameBtn">Start Game</button>
        </div>
    </center>
</div>
```

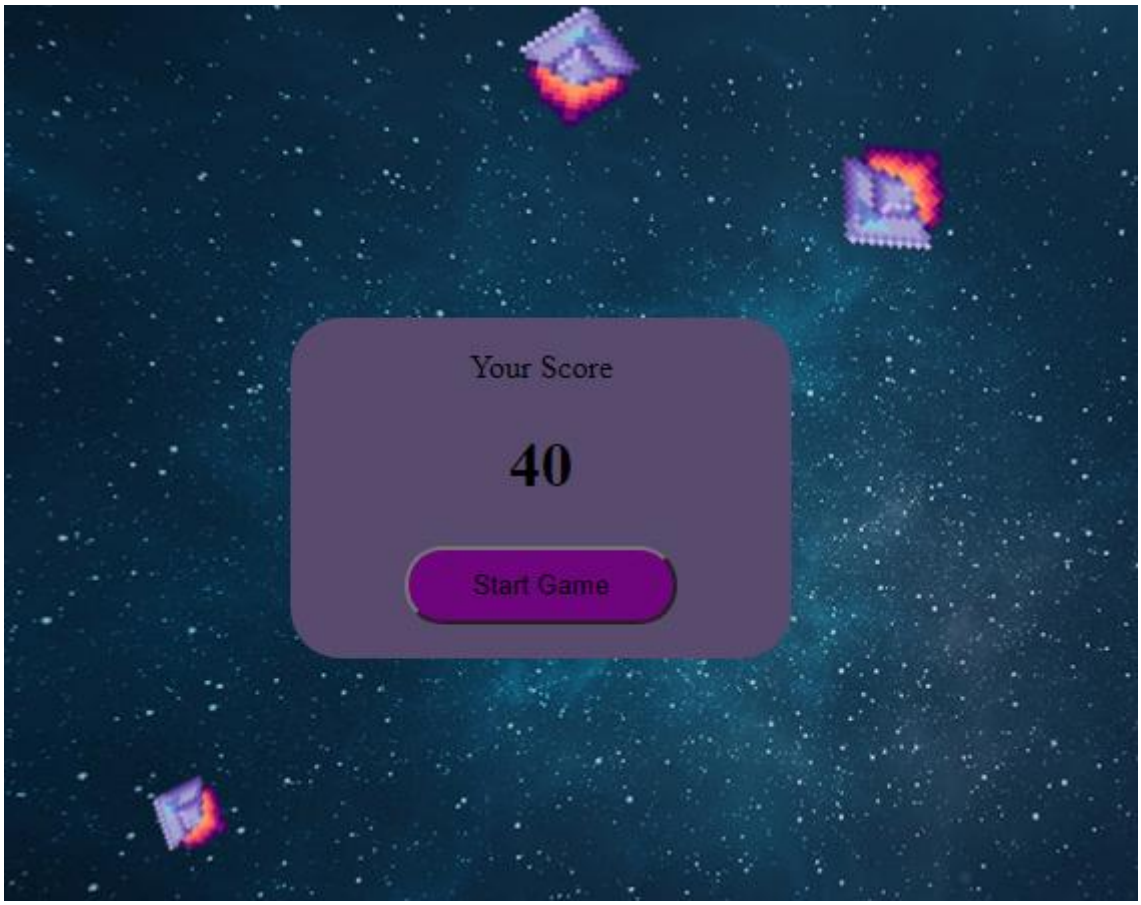
Rys.4.1.1 Okno dialogowe w HTML

Które po prostu przestaje być wyświetlane po naciśnięciu przycisku „Start Game”. Równocześnie zostaje utworzony drugi obiekt Canvas, który teraz jest widoczny dla użytkownika. W tym obiekcie wyświetlana jest gra.

```
startGameBtn.addEventListener('click',() => {
    init();
    animate();
    GameOverEl.style.display = 'none'
})
```

Rys.4.1.2 Wyłączenie okna dialogowego

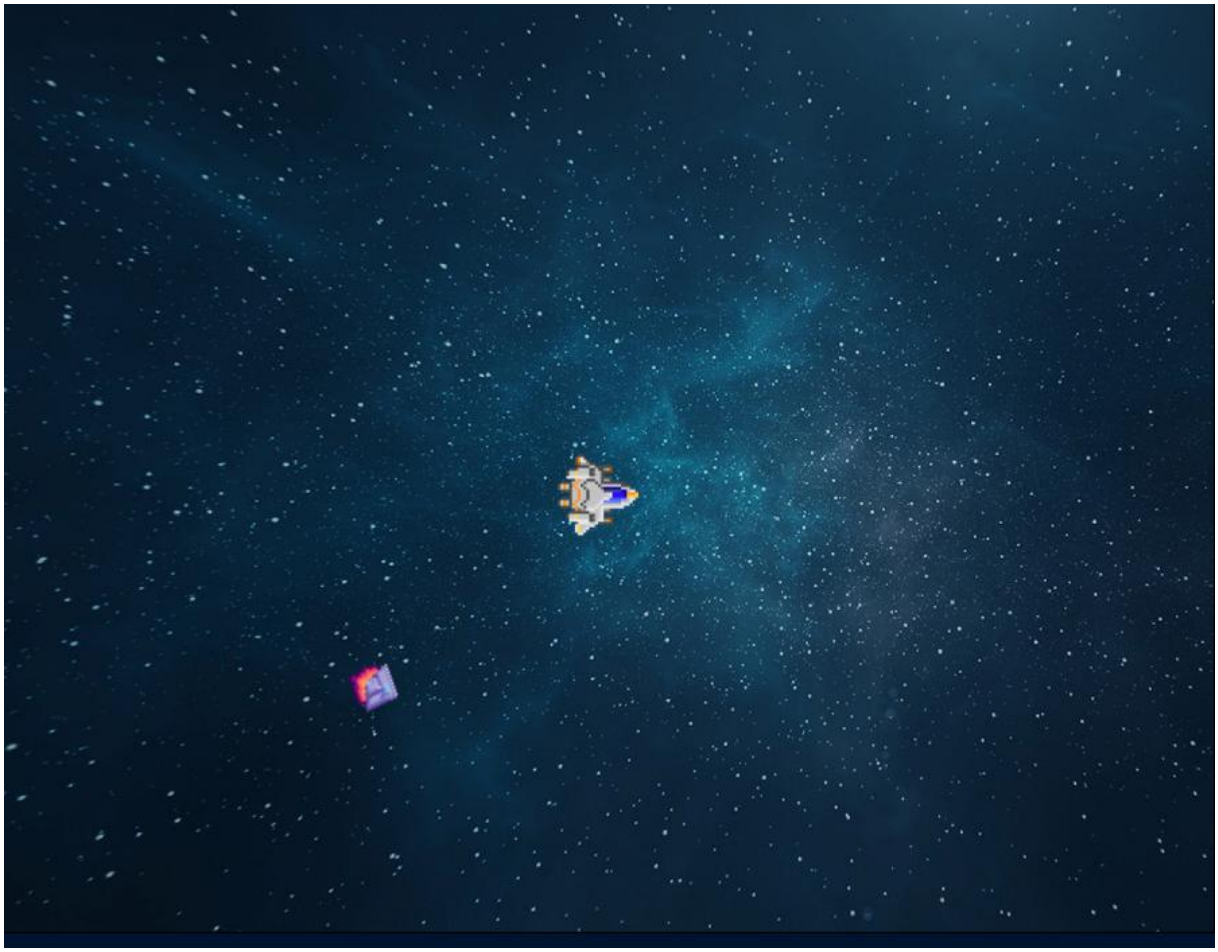
Te samo okno dialogowe zostaje wyświetlone, gdy punkty zdrowia gracza spadną do zera, zostaną wtedy wyświetlone punkty, które udało się zdobyć podczas danej gry.



Rys.4.1.3 Okno dialogowe końca gry

3.2 Zasada działania gry.

Gracz steruje statkiem kosmicznym, a jego zadaniem jest bronić się przed nieskończoną falą wrogich okrętów. Gra polega na zdobyciu jak najwyższej liczby punktów zanim gracz utraci wszystkie punkty zdrowia. Całość gry jest napisania w Java Script.



Rys.4.2 Rozpoczęta gra

3.3 Obiekt gracza.

Wraz z rozpoczęciem gry tworzone jest kilka grup obiektów. Pierwszym stworzonym obiektem jest gracz, który zostaje ustawiony na środku elementu Canvas.

Rys.4.3.1 Gracz

Konstruktor odgórnie przypisuje mu następujące parametry.

```
constructor(x,y,dx,dy,radius) {
this.x = x; //Położenie względem osi x
this.dx = dx; // Przesunięcie względem osi x
this.y = y; //Położenie względem osi y
this.dy = dy; // Przesunięcie względem osi y
this.radius = radius; // Średnica
this.topspeed = 3; // Maksymalna prędkość
this.acc = 0.05; // Przyśpieszenie
this.phealth = 100; // Punkty Zdrowia
this.points = 0; // Punkty
this.angle = 0; // Obrót
}
```

Rys.4.3.2 Konstruktor obiektu gracz

- Przesunięcie to wartość, która zostaje dodana do obecnego położenia, by utworzyć wrażenie ruchu gracza.
- Średnica – Poszczególne obiekty są oparte na bazie okręgu, dlatego rozmiar każdego z nich określa się średnicą
- Przyśpieszenie – ruch w grze odbywa się przy użyciu klawiszy W,S,A,D. Czym dłużej wciśnięty jest dany przycisk tym szybciej gracz będzie się poruszał w danym kierunku. Gracz będzie przyśpieszał do momentu osiągnięcia maksymalnej prędkości lub do momentu puszczenia odpowiedniego przycisku i będzie

utrzymywał osiągniętą prędkość dopóki nie zostanie wciśnięty przycisk ruchu w przeciwnym kierunku.

```
movement = function(){


    window.addEventListener("keydown", function(e){
        keys[e.keyCode] = true;
    });

    window.addEventListener("keyup", function(e){
        delete keys[e.keyCode];
    });

    if(keys[87] && this.dy >- this.topspeed){
        this.dy -= this.acc;
    }
    if(keys[83] && this.dy < this.topspeed){
        this.dy += this.acc;
    }
    if(keys[65] && this.dx > -this.topspeed){
        this.dx -= this.acc;
    }
    if(keys[68] && this.dx < this.topspeed){
        this.dx += this.acc;
    }

}
```

Rys.4.3.3 Ruch gracza kierowany odpowiednimi przyciskami W,S,A,D

- Punkty zdrowia – gdy ta wartość spadnie do 0, gra zostanie zakończona
- Punkty – Licznik zdobytych przez gracza punktów, które zostaną wyświetlone po zakończeniu gry.
- Obrót – Model gracza obraca się w stronę wskaźnika myszki na ekranie. Jest tu zapisywana wartość wyrażona w stopniach, która mówi o ile ma zostać obrócony gracz.

3.3.1 Rysowanie gracza.

Rysowanie gracza ogranicza się do rysowania zawartej grafiki na danych koordynatach. Problem się pojawia gdy tą grafikę należy obrócić.

```
draw = function(){

    var upx = this.x;
    var upy = this.y;
    var upr = this.radius;

   if(playerloaded){
   c.save();
   c.translate(upx,upy);
   c.rotate(this.angle);
   c.drawImage(imgPlayer,-upr,-upr,upr*2,upr*2);
   c.restore();
   }

}
```

Rys.4.3.1.1 Rysowanie gracza

Żeby obrócić gracza najpierw należy zapisać stan całego elementu Canvas. Następnie przenosi się element Canvas na obecne położenie gracza, ponieważ funkcja obracająca obraz ‘rotate()’ obraca je o żądany kąt wobec koordynatu (0,0), który jest lewym górnym rogiem elementu Canvas.

Kąt wyliczany jest przy pomocy formuły :

arcus tangens (pozycja x kursora – pozycja x gracza,

-(pozycja y kursora – pozycja y gracza))

Funkcja Draw rysuje gracza, a funkcja ‘restore()’ przywraca stan całego elementu Canvas do ostatnio zapisanej jego instancji.

Ważnym też jest, by gracz nie mógł opuścić miejsca gry. Dba o to funkcja, która sprawdza, czy gracz nie wyszedł poza granice elementu Canvas.

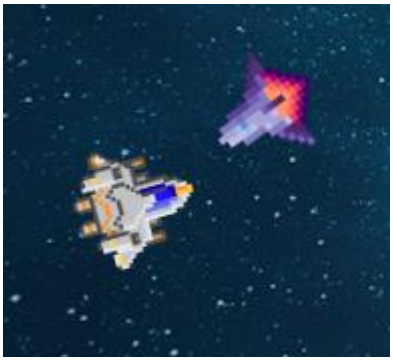
```
borderchеck = function(){

    var stepAway = 1;

    if( this.x-this.radius  < 0){
        this.dx = 0;
        this.x = this.radius+stepAway;
    }

    if(this.x+this.radius  > canvas.width){
        this.dx = 0;
        this.x = canvas.width - (this.radius+stepAway);
    }

    if(this.y+this.radius > canvas.height ){
        this.dy = 0;
        this.y = canvas.height - (this.radius+stepAway);
    }

    if(this.y-this.radius < 0){
        this.dy = 0;
        this.y = this.radius+stepAway;
    }
}
```

Rys.4.3.1.2 Funkcja uniemożliwiająca opuszczenie obszaru gry

Dla każdej krawędzi ekranu sprawdzane jest czy aktualna pozycja gracza nie wychodzi poza wysokość lub szerokość elementu Canvas. Gdy do tego dojdzie prędkość gracza jest ustawiana na ‘0’ i gracz zostaje cofnięty o ‘1’ jednostkę w tył.

3.4 Antagoniści

W grze istnieją dwa typy wrogów, z którymi graczowi przyjdzie się zmierzyć. Pierwszym typem przeciwnika jest ‘The Follower’ , który podąża za graczem dopóki w niego nie uderzy.



Rys.4.4.1 Follower podążający za graczem

Przeciwnik ten również się obraca jak statek gracza przy użyciu tej samej formuły na obliczenie kąt obrotu tylko zamiast być skierowanym w stronę kursora jest skierowany w stronę pozycji gracza. Przyśpieszenie z jakim porusza się ten obiekt zostaje wyliczone w następujący sposób :

Velocity = {x: Math.cos(angle),Math.sin(angle)}

Drugim typem przeciwnika jest ‘Small Cruiser’ ,który przelatuje w linii prostej przez plansze strzelając do gracza.

Rys.4.4.2 Small Cruiser przelatujący przez planszę.

‘Small Cruiser’ jest obrócony przodem w stronę w którą podąża.

3.4.1Pojawianie się przeciwników

Typ przeciwnika, który się pojawia jest decydowany poprzez sprawdzenie, czy wygenerowana losowa liczba przekracza dany próg liczbowy. Odpowiada za to funkcja ‘SpawnEnemies()’

```
function SpawnEnemies(){

    let bet;
        const radius = 10 + Math.random()*25;
        let x
        let y

        if(Math.random() < 0.5 ){
         x = Math.random() < 0.5 ? 0 - radius : canvas.width + radius
         y = Math.random()*canvas.height
        } else {
             y= Math.random() < 0.5 ? 0 - radius : canvas.height + radius
             x= Math.random()*canvas.width
        }

        const angle = Math.atan2(Math.random()*(canvas.height) - y,  Math.random()*canvas.width - x)

        var velocity ={
            x: Math.cos(angle),
            y: Math.sin(angle)
        }

    bet = Math.random();

    if(bet>0.1){
    enemyArray.push(new SmallC(x,y,radius,velocity));
    }else{
    enemyArray.push(new Follower(x,y,radius,velocity));
    }

}
```

Rys.4.4.1.1 Funkcja SpawnEnemies.

Wrogowie pojawiają się w losowym miejscu poza granicami planszy gry i zostają zapisani w tablicy wrogów. W przypadku typu przeciwnika ‘Small Cruiser’ musi zostać wybrany kierunek, w którym będzie on skierowany. Wybrany kierunek lotu jest determinowany poprzez wybrania losowego punktu, który znajduje się na planszy gry.

Gdy wróg znajdzie się poza planszą gry jest on usuwany z tablicy.

```
function isOutf(obj){
    if (obj.x  < -200 || obj.x  > canvas.width + 200 ||
        obj.y  < -200 || obj.y  > canvas.height + 200 ){
            isOut = true;
        }

}
```

Rys.4.4.1.2 Funkcja sprawdzająca czy wróg znajduje się poza granicami gry

Warunki sprawdzające czy przeciwnik jest poza granicami gry mają rozszerzone granice planszy gry, by uniknąć usuwana oponentów, którzy dopiero się pojawili.

3.5 Walka

‘Small Cruiser’ oraz gracz mają możliwość strzelania pociskami. ‘Small Cruiser’ zawsze strzela w aktualną pozycję gracza w momencie, w którym wystrzeliwuje pocisk. Pociski gracza zawsze zmierzają w stronę pozycji kursora w momencie wystrzelenia pocisku.

Klasa Projectile odpowiada za wszystkie wystrzelone pociski.

```
class Projectile {

    constructor(x,y,radius,velocity,type) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.velocity = velocity;
        this.type = type;       //0- friendly 1- enemy
        }

    draw(){
        c.beginPath()
        c.arc(this.x,this.y,this.radius, 0, Math.PI * 2, false)
        if(this.type == 0){
            c.fillStyle = "white";
        }else{
            c.fillStyle = "red";
        }
        c.fill();

    }

    update() {
        this.draw();
        this.x = this.x +this.velocity.x;
        this.y = this.y + this.velocity.y;
    }
}
```

Rys.4.5.1 Klasa Projectile

Pociski są rozróżniane ze względu na to przez kogo zostały wystrzelone by uniknąć sytuacji gdzie jeden z przeciwników trafi innego przeciwnika.

Chybione pociski mogą lecieć w nieskończoność poza ustaloną planszą gry, dlatego w momencie gdy wyjdą poza granicę gry zostają one usunięte z tablicy pocisków.

```
function projectileremover(){
    projectiles.forEach((projectile,pindex)=>{
        projectile.update();
        //removing projectiles
        if (projectile.x - projectile.radius < 0 || projectile.x + projectile.radius > canvas.width ||
            projectile.y - projectile.radius < 0 || projectile.y  + projectile.radius > canvas.height ){
            setTimeout(()=>{

                projectiles.splice(pindex, 1)
                },0)
        }
    })
}
```

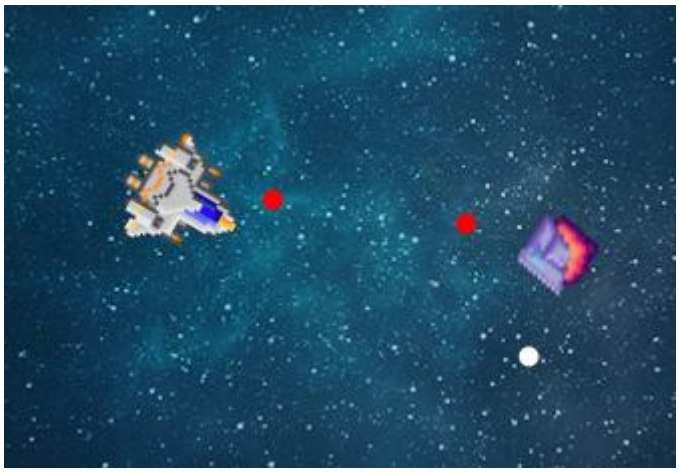
Rys.4.5.2 Funkcja usuwająca pociski

Funkcja ‘gamelooper()’ odpowiada za cały system walki w grze. Sprawdza czy pocisk trafił w, zależnie od typu, przeciwnika lub gracza. Jeżeli pocisk gracza trafi w przeciwnika, dany przeciwnik zostaje usunięty z gry ,a gracz zostaje nagrodzony punktami.

Jeżeli pocisk wystrzelony przez przeciwnika trafi w gracza, gracz traci punkty zdrowia.

Jeżeli statek gracza i statek przeciwnika zderzą się, statek przeciwnika zostaje usunięty, a statek gracza traci punkty zdrowia.

Gdy punkty zdrowia gracza w wyniku tych zdarzeń zejdą do 0, gra zostaje zakończona



Rys.4.5.3

Każdy pokonany przeciwnik ma małą szanse no zostawienie po sobie zestawu naprawczego, który po podniesieniu przywraca zdrowie gracza do 100 punktów.

Rys.4.5.3 Zestaw naprawczy

5. Podsumowanie

Ostatecznie ukończenie tej pracy nauczyło mnie pisania w języku JavaScript, a także korzystania z elementu Canvas. Dowiedziałem się też sporo o tworzeniu grafik oraz animacji na stronach HTML. Projekt planuje dalej rozwijać i rozszerzać o nową zawartość. Chciałbym zwiększyć ilość przeciwników, dodać przeciwników specjalnych oraz rozszerzyć asortyment z którego może korzystać gracz. Same mechaniki zaimplementowane już w tym projekcie będę w stanie użyć ponownie w innych podobnych projektach.

6. Literatura

https://www.w3schools.com/html/html5_canvas.asp

https://developer.mozilla.org/pl/docs/Web/API/Canvas_API/Tutorial

https://itnext.io/building-a-space-shooter-game-with-html5-canvas-typescript-part-3-f1b2808e85bd