

Projekt - Gra Mapa

Osobny plik JavaScript

Dla zachowania czytelności projektu, cały kod JavaScript będzie przechowywany w oddzielnym pliku game.js. Skrypt pomiędzy znacznikami <script> i </script> z pliku HTML należy przenieść do nowego pliku JS i umieścić w funkcji wywoływanej, kiedy został już wczytany HTML i zostało stworzone drzewo dokumentu.

```
game.js
document.addEventListener('DOMContentLoaded', function(event) {
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');

    canvas.width = 800;
    canvas.height = 600;

    context.fillStyle = '#e0e0e0';
    context.fillRect(0, 0, canvas.width, canvas.height);

    console.log('DONE');
})
```

Więcej na temat metody addEventListener w JavaScript można przeczytać na stronie https://www.w3schools.com/jsref/met_document_addeventlistener.asp.

Więcej na temat narzędzi dla programistów można przeczytać na stronach <https://support.google.com/displayvideo/answer/6188066?hl=pl> oraz https://www.w3schools.com/jsref/met_console_log.asp.

Teraz plik HTML powinien zawierać tylko ładowanie pliku game.js.

```
index.html
<!DOCTYPE html>
<html>

    <head>
        <title>GRA</title>
        <link rel="stylesheet" type="text/css" href="style.css" />
        <script type="text/javascript" src="game.js"></script>
    </head>

    <body>
        <canvas id="canvas"></canvas>
    </body>

</html>
```


Poziom gry jako klasa

W grze może wystąpić kilka instancji poziomu gry, ale struktura każdego takiego poziomu powinna być niezmienna. W związku z tym tworzona jest podstawowa klasa Level z konstruktorem, w którym definiowane są wszystkie zmienne opisujące poziom. Do konstruktora przekazywane są wartości:

- wallWidth i wallHeight - szerokość i wysokość ściany ustawiana dla parametru wall,
- canvasWidth i canvasHeight - wymiary elementu canvas wykorzystywane do wyliczenia wartości parametrów width i height,
- map - mapa zdefiniowana jako tablica dwuwymiarowa z definicją ułożenia typów ścian na poziomie.

Rodzaje ścian są zdefiniowaneściśle w konstruktorze w zmiennej walls, ponieważ zakładamy, że każdy poziom posiada te same rodzaje ścian. Jeżeli założenie jest odwrotne, należyzmodyfikować konstruktor tak, aby przyjmował kolejną wartość i przypisywał ją do parametru walls.

```
game.js
class Level {
    constructor(wallWidth, wallHeight, canvasWidth, canvasHeight, map) {
        // rozmiar ściany
        this.wall = { width: wallWidth, height: wallHeight };

        // rozmiar poziomu
        this.width = canvasWidth / wallWidth;
        this.height = canvasHeight / wallHeight;

        // rodzaje ścian
        this.walls = [
            {id: 'background', colour: '#e0e0e0', solid: 0 },
            {id: 'wall', colour: '#a0a0a0', solid: 1 },
            {id: 'win', colour: '#00ff00', solid: 0 },
            {id: 'lose', colour: '#ff0000', solid: 0 }
        ];
        // mapa
        this.map = map;
    }
};
```

Więcej na temat klas w JavaScript można przeczytać na stronie
https://www.w3schools.com/js/js_classes.asp.

Sama definicja instancji klasy Level pozostaje w funkcji zdarzenia DOMContentLoaded. Do konstruktora zostają przekazane wyżej wspomniane już wymiary ściany (każdy student powinien podać własną wartość), wymiary elementu canvas oraz wcześniej zdefiniowana (jako zmienna) mapa.

```
game.js
level = new Level(25, 25, canvas.width, canvas.height, map);
```

Rysowanie mapy

Wizualizacja poziomu odbywa się poprzez rysowanie każdej kolejnej ściany zdefiniowanej w tablicy dwuwymiarowej `map`. Przechodząc przez całą mapę, wyznaczane są kolejno:

- pierwszy punkt skrajny ściany na podstawie aktualnej pozycji rysowanej ściany oraz wymiarów ściany,
- drugi punkt skrajny ściany na podstawie aktualnej pozycji rysowanej ściany oraz wymiarów ściany,
- kolor rysowanej ściany.

Zawartość elementu `canvas` także jest przekazywana, ponieważ klasa `Level` nie ma do niego bezpośredniego dostępu.

```
game.js
drawMap(context) {
    for (var x=0; x<level.width; x++) {
        for (var y=0; y<level.height; y++) {
            this.drawWall(
                x*this.wall.width,
                y*this.wall.height,
                (x+1)*this.wall.width,
                (y+1)*this.wall.height,
                this.walls[this.map[y][x]].colour,
                context
            );
        }
    }
};
```

Metoda `drawMap(context)` korzysta z metody `drawWall(x1, y1, x2, y2, colour, context)`, zatem następnym krokiem jest zdefiniowanie tejże metody w klasie `Level`. Metoda odpowiada za rysowanie prostokąta o punktach skrajnych `(x1, y1)` i `(x2, y2)` w kolorze `colour`. To jest zadanie do samodzielnego wykonania przez studenta.

Ostatnim krokiem jest wywołanie w funkcji zdarzenia `DOMContentLoaded` metody rysującej określony poziom w elemencie `canvas`.

```
game.js
level.drawMap(context);
```

Kontrola nad wartościami

Aby móc wprowadzić kontrolę nad wartościami zmiennych opisujących poziom, należy zmienić typ parametrów klasy z publicznego na prywatny. W tym celu należy zastosować poniższe wymagania:

- Zmienna prywatna zawsze przyjmuje przed swoją nazwą symbol `#`.
- W klasie musi znaleźć się definicja parametrów. W konstruktorze przypisywane są tylko wartości do tych parametrów.
- W klasie muszą znaleźć się metody `set` odpowiedzialne za ustawianie wartości parametrów.
- W klasie muszą znaleźć się metody `get` odpowiedzialne za dostęp do parametrów.

W poniższym kodzie zmieniono publiczny parametr `wall` na prywatny. Pojawiła się definicja prywatnej zmiennej. Wprowadzono także metodę `get` oraz `set`, w której to jednocześnie uaktualniane są wartości parametrów `width` i `height`, ponieważ zależą one od wymiarów ściany. Teraz należy uczynić to samo z parametrami `width`, `height`, `walls` i `map`, przy czym należy zastanowić się, które zmienne powinny mieć metodę `set`, a które metodę `get`. Należy także poprawić metody `drawMap(...)` oraz `drawWall(...)`, aby korzystały z prywatnych parametrów klasy. To jest zadanie do samodzielnego wykonania przez studenta.

```
game.js
class Level {
    #wall;
    [...]

    constructor(wallWidth, wallHeight, canvasWidth, canvasHeight, map) {
        // rozmiar ściany
        this.#wall = { width: wallWidth, height: wallHeight };
        [...]
    }

    set wall(size) {
        var canvasWidth = this.#wall.width * this.#width;
        var canvasHeight = this.#wall.height * this.#height;

        this.#wall = { width: size.width, height: size.height };

        this.#width = canvasWidth / size.width;
        this.#height = canvasHeight / size.height;
    }
    [...]

    get wall() {
        return this.#wall;
    }
    [...]

    drawWall(x1, y1, x2, y2, colour, context) {
        [...]
    };

    drawMap(context) {
        [...]
    };
}
```

Sprawdzanie poprawności mapy

Poprawna mapa poziomu powinna przechowywać taką liczbę ścian, jaki jest zdefiniowany wymiar poziomu. Do weryfikacji mapy posłuży nowa metoda `checkMap(map)` w klasie `Level`. Metoda najpierw znajduje najmniejszą liczbę ścian poziomo i pionowo, a następnie uzyskany wynik porównuje z szerokością i wysokością poziomu. W zależności od rezultatu zwracana jest wartość `true` lub `false`.

```
game.js
checkMap(map) {
    var minWidth = map[0].length;
    var minHeight = map.length;
    for (var y=1; y<this.#height; y++) {
        if (map[y].length < minWidth) minWidth = map[y].length;
    }

    if (minWidth == this.#width && minHeight == this.#height) return true;
    else return false;
}
```

Teraz należy poprawnie wywołać funkcję `checkMap(map)` we wszystkich miejscach, gdzie mapa jest ustawiana lub zmieniana. W przypadku zwróconej wartości `false` przez funkcję `checkMap(map)`, należy wywołać komunikat na stronie - `alert()`. To jest zadanie do samodzielnego wykonania przez studenta.

Więcej na temat metody `alert` w JavaScript można przeczytać na stronie https://www.w3schools.com/jsref/met_win_alert.asp.

Na koniec dodać do funkcji `checkMap(map)` warunek sprawdzający, czy mapa zawiera dozwolone wartości, czyli liczby reprezentujące zdefiniowane rodzaje ścian. Warto tutaj użyć funkcji `length` dla tablic. To jest zadanie do samodzielnego wykonania przez studenta.

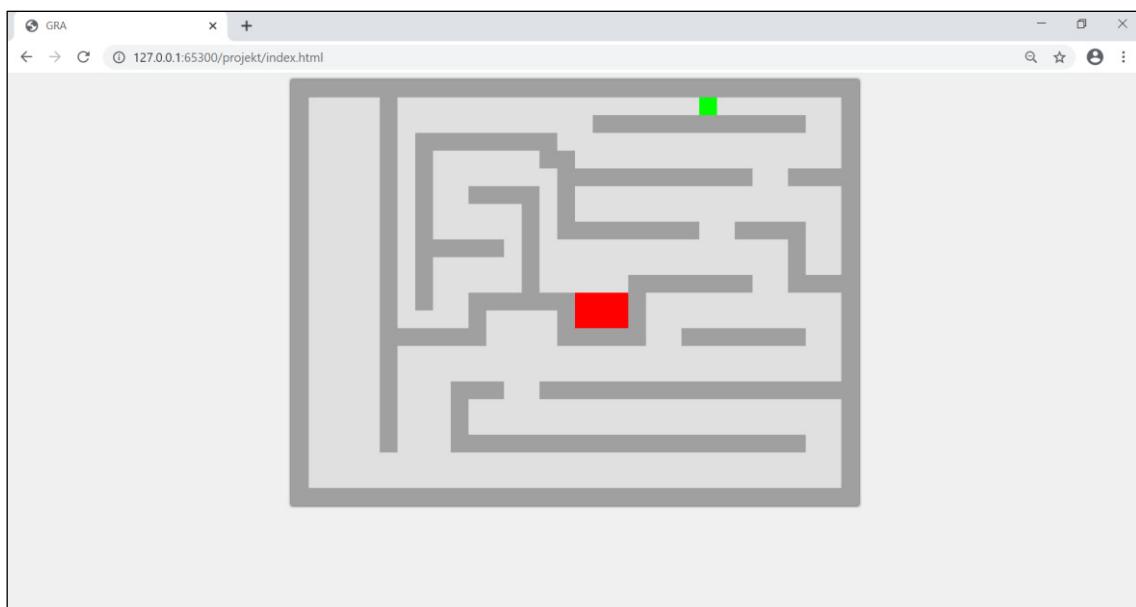
Sprawdzanie poprawności wymiaru ściany

Zdefiniować metodę w klasie `Level` sprawdzającą, czy szerokość i wysokość ściany nie jest zbyt duża w porównaniu do rozmiaru elementu `canvas`. Można przyjąć dowolny warunek, np. że ściana musi mieć co najmniej 3krotnie mniejsze wymiary niż `canvas`. To jest zadanie do samodzielnego wykonania przez studenta.

Metodę zastosować we wszystkich miejscach, gdzie wymiary ściany są ustawiane lub zmieniane. To jest zadanie do samodzielnego wykonania przez studenta.

Efekt końcowy

Po ukończeniu tej części, projekt powinien wyświetlać zdefiniowaną mapę z 4 rodzajami ścian. Poniżej Rysunek 1 jako przykład.



Rysunek 1. Efekt końcowy treści przedstawionej powyżej.