Projekt semestralny
Gra „Pole Minowe”

Wykonawcy:

Paweł Korzec, Łukasz Cegielski

Poznań 2024

Spis treści

1. Założenia projektu..... 4
2. Opis funkcjonalności..... 5
3. Instrukcja obsługi gry..... 21
4. Kod źródłowy gry..... 23
5. Wnioski końcowe..... 52

Spis ilustracji

1. Funkcja startgame..... 5
2. Funkcja startgameSound..... 5
3. Funkcja Ustawienia..... 6
4. Funkcja UstawieniaSound..... 6
5. Funkcja zakoncz..... 6
6. Funkcja zakonczSound..... 6
7. Funkcja fdiff..... 7
8. Funkcja fsize..... 8
9. Funkcja customsize..... 8
10. Funkcja customdiff..... 9
11. Funkcja save..... 9
12. Funkcja back..... 10
13. Funkcja backSound..... 10
14. Funkcja sprawdzpole..... 11
15. Funkcja sprawdzpoladookola..... 11
16. Funkcja odkryjpole..... 12
17. Funkcja losnum..... 13
18. Funkcja wys..... 14
19. Funkcja conf..... 15
20. Funkcja timeout..... 15
21. Funkcja Lose..... 16
22. Nienazwana funkcja..... 16
23. Funkcja Funkcja tenseconds oraz nienazwana funkcja..... 17
24. Funkcja koniecgry..... 17
25. Funkcja oflaguj..... 18
26. Funkcja convertTime..... 19
27. Funkcja Nienazwana funkcja 2..... 20
28. Funkcja back 2..... 20
29. Funkcja backSoud..... 20
30. Odkryte bezpieczne pola..... 21

31. Rozbrajacz..... 21
32. Rozbrojona mina..... 22
33. Flagi..... 22

Spis kodów źródłowych

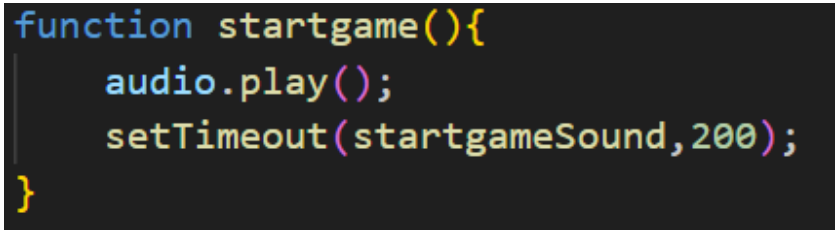
scriptindex.js..... 23
scrtiptsettings.js..... 24
script.js..... 30
endpg.js..... 49

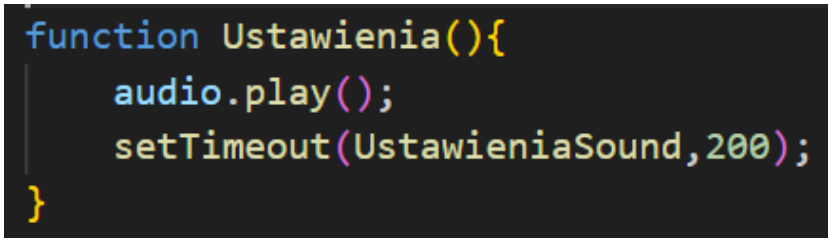
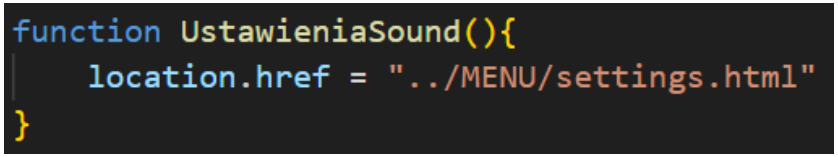
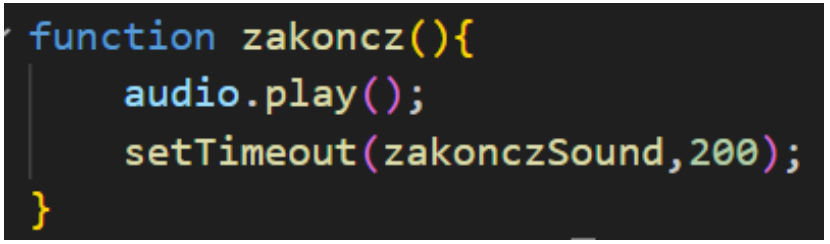
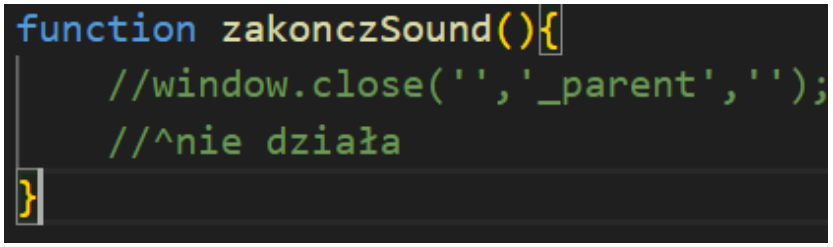
1. Założenia projektu

Celem naszego projektu było dostarczenie działającej wersji gry “saper” jako aplikacja internetowa. Chcieliśmy żeby gra miała proste, przejrzyste i schludne UI, pasujące dźwięki i muzykę, opcje tworzenia własnych plansz i poziomów trudności oraz przegląd statystyk gracza po zakończeniu rozgrywki

2. Opis funkcjonalności

MENU/scriptindex.js:

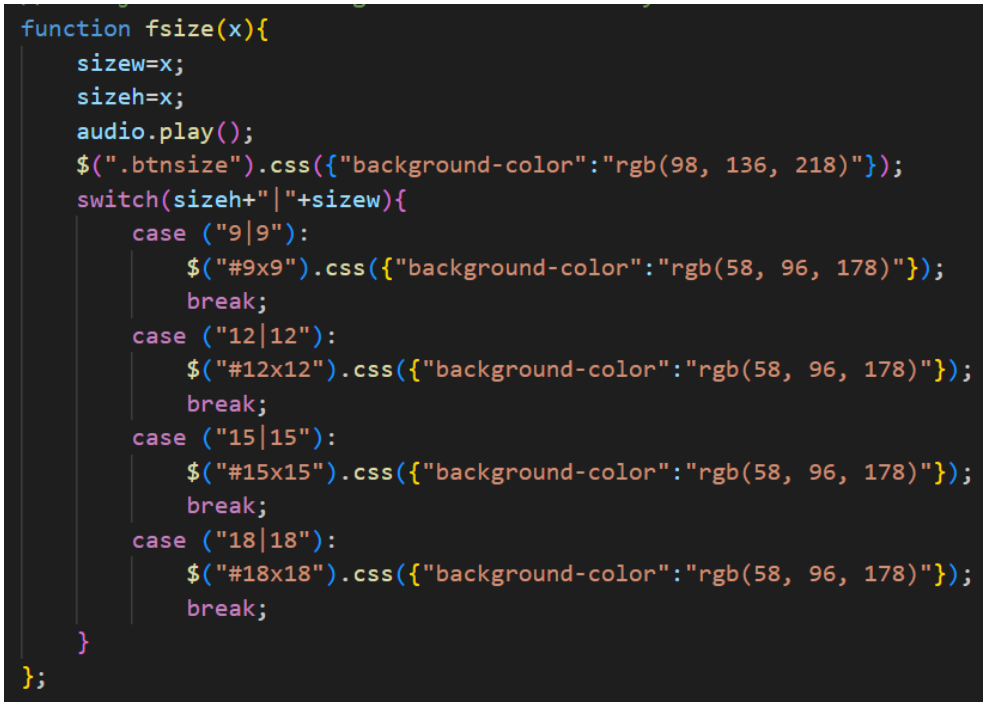
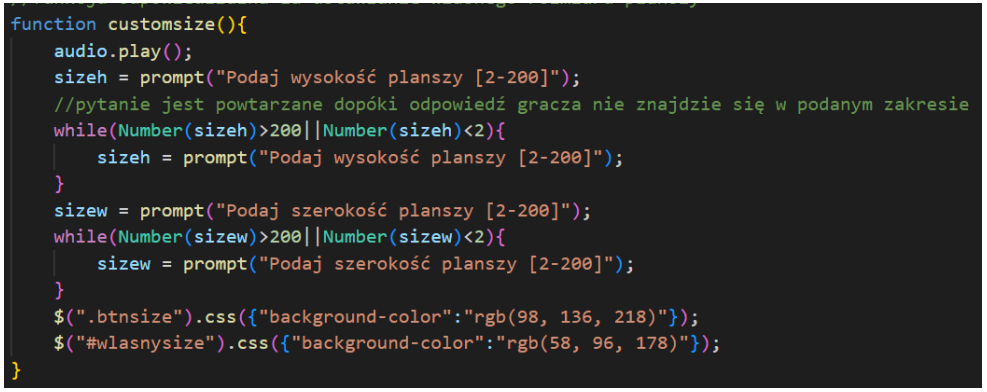
Nazwa Funkcji JavaScript	**Opis działania**	**Kod**
startgame	funkcja gra odpowiedni plik dźwiękowy następnie wywołuje funkcję startgameSound. Podzielenie tej oraz kilku innych funkcji na wersję zwykłą oraz ‘sound’ okazało się niezbędne jeżeli chcieliśmy żeby każdy dźwięk grał bez przeszkód.	 rys.1 Funkcja startgame
startgameSound	funkcja sprawdza czy ustawienia były zmieniane. Jeżeli nie, ustawia domyślne. Następnie, funkcja przerzuca gracza z menu do gry.	 rys.2 Funkcja startgameSound

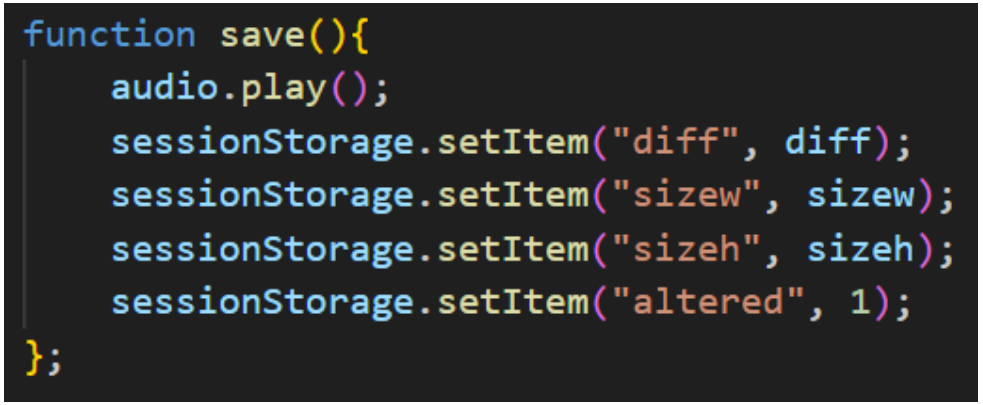
Ustawienia	funkcja gra odpowiedni plik dźwiękowy następnie wywołuje funkcję UstawieniaSound.	 rys.3 Funkcja Ustawienia
UstawieniaSound	Funkcja przerzuca gracza na stronę z ustawieniami	 rys.4 Funkcja UstawieniaSound
zakoncz	Funkcja gra odpowiedni plik dźwiękowy następnie wywołuje funkcję zakonczSound.	 rys.5 Funkcja zakoncz
zakonczSound	Funkcja miała zamykać kartę gry co jest niemożliwe dla kart otwartych przez użytkownika nie stronę. W obecnym stanie funkcja nie działa z powodu limitów nałożonych przez przeglądarki.	 rys.6 Funkcja zakonczSound

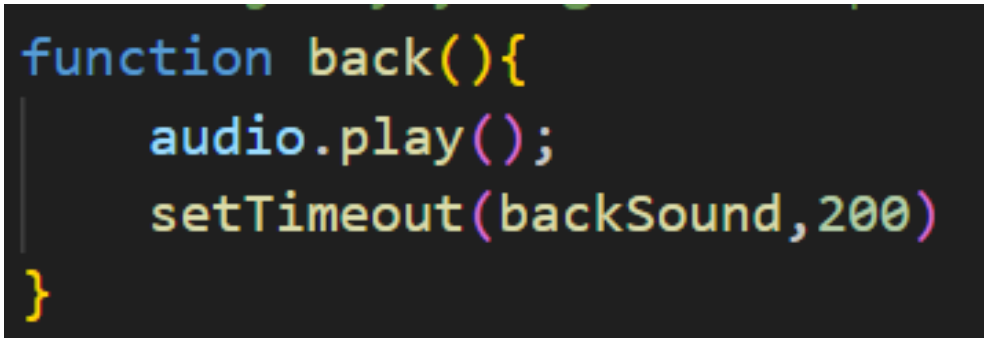
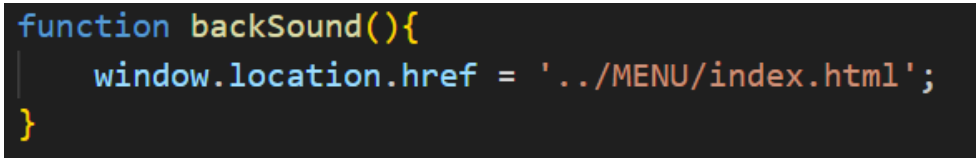
MENU/scriptsttings.js:

Nazwa funkcji JavaScript	**Opis działania**	**Argumenty**	**Kod**
fdiff	Funkcja zmienia kolor guzików, odtwarza odpowiedni plik dźwiękowy i ustawia nową trudność.	x - wybrana trudność, procent zaminowanych pól	 rys.7 Funkcja fdiff

```
function fdiff(x){
    diff=x;
    audio.play();
    $(".btndiff").css({"background-color":"rgb(98, 136, 218)"});
    switch(diff){
        case 10:
            $("#easy").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case 20:
            $("#normal").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case 30:
            $("#hard").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case 40:
            $("#impossible").css({"background-color":"rgb(58, 96, 178)"});
            break;
    }
};
```

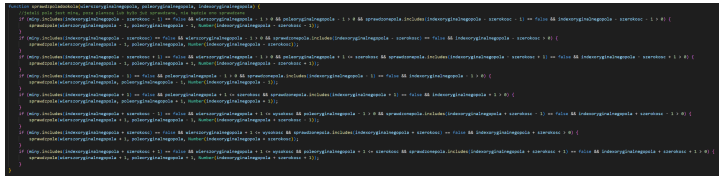
fsize	Odpowiednik funkcji fdiff dla rozmiaru planszy	x - wybrana szerokość i wysokość planszy	 rys.8 Funkcja fsize
customsize	Funkcja podobna w działaniu do funkcji fsize lecz zamiast wybierać spośród podanych rozmiarów użytkownik sam podaje wymiary które sam wybrał		 rys.9 Funkcja customsize

customdiff	Odpowiednik funkcji customsize dla poziomu trudności		 rys.10 Funkcja customdiff
save	Funkcja gra odpowiedni plik dźwiękowy następnie zapisuje wybrane ustawienia i fakt że były one zmieniane		 rys.11 Funkcja save

back	Funkcja gra odpowiedni plik dźwiękowy następnie wywołuje funkcję backSound.		 rys.12 Funkcja back
backSound	Funkcja przerzuca gracza z powrotem do menu		 rys.13 Funkcja backSound

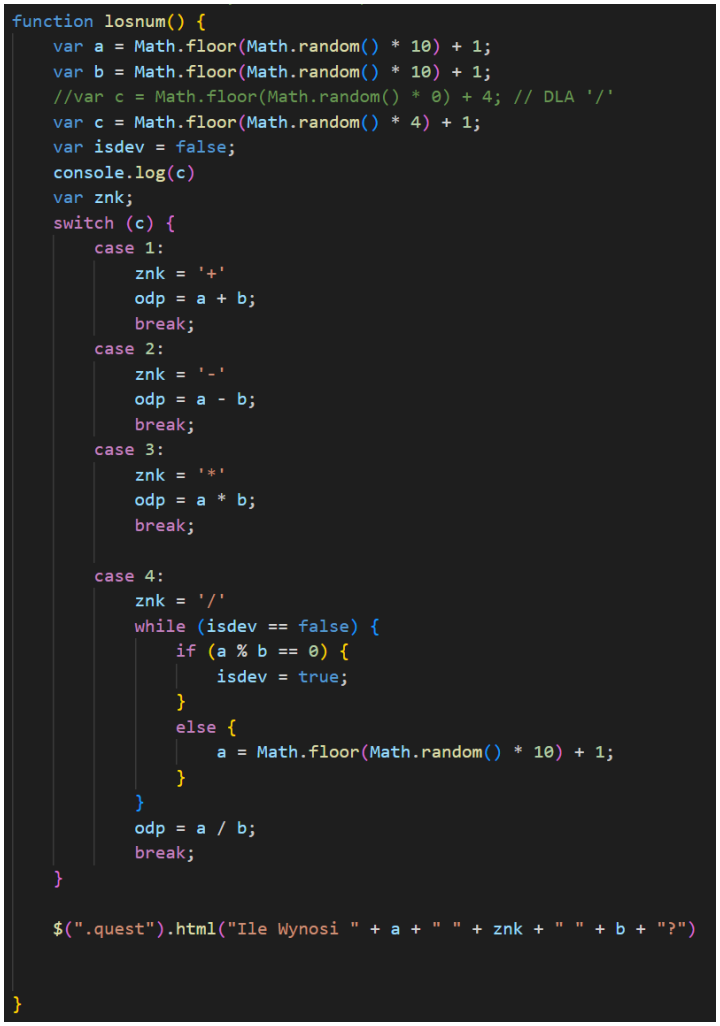
PLANSZA/script.js:

Nazwa funkcji JavaScript	**Opis działania**	**Argumenty**	**Kod**

sprawdzpole	Jeżeli pole nie było już sprawdzane, funkcja odtwarza odpowiedni dźwięk i liczy ile pól dookoła jest minami. Następnie, funkcja dodaje numer pola do listy sprawdzonych pól, odświeża licznik min i wywołuje funkcję odkryjpole. Jeżeli ilość min dookoła pola wyniosła 0 funkcja wywołuje funkcję sprawdzpoladookola. Ostatecznie, jeżeli nie ma pól które nie były już sprawdzone i nie są minami, funkcja wywołuje funkcję koniecgry.	wierszwybranegopola - numer wierszu wybranego pola polewybranegopola - numer pola wybranego pola w wierszu (numer kolumny wybranego pola) indexwybranegopola - numer pola liczony od góry do dołu i od lewej do prawej	 rys.14 Funkcja sprawdzpole
sprawdzpoladookola	Funkcja sprawdza czy pola dookoła podanego pola znajdują się na planszy, nie były już sprawdzane i nie są minami. Każde pole które spełnia te warunki jest następnie	wierszoryginalnegopola - numer wierszu oryginalnego pola poleoryginalnegopola - numer pola oryginalnego pola w wierszu (numer kolumny oryginalnego pola) indexoryginalnegopola - numer oryginalnego pola liczony od góry do dołu i od prawej do lewej	 rys.15 Funkcja sprawdzpoladookola

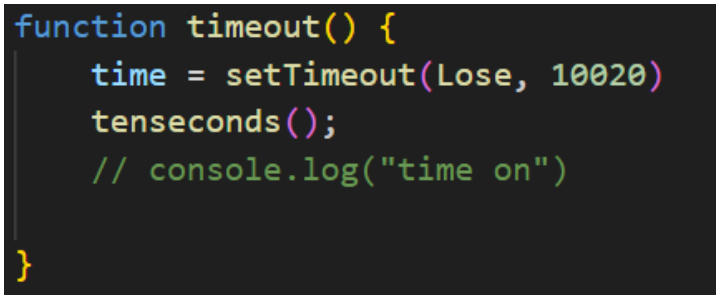
	sprawdzane funkcją sprawdzpole.		
odkryjpole	Funkcja wypisuje ilość otaczających min na polu w odpowiednim kolorze	wierszdoodkrycia - numer wiersza pola do odkrycia poledoodkrycia - numer pola w wierszu do odkrycia (numer kolumny pola do odkrycia) minydookolaodkrywanegopola - ilość min dookoła odrywanego pola	 rys.16 Funkcja odkryjpole

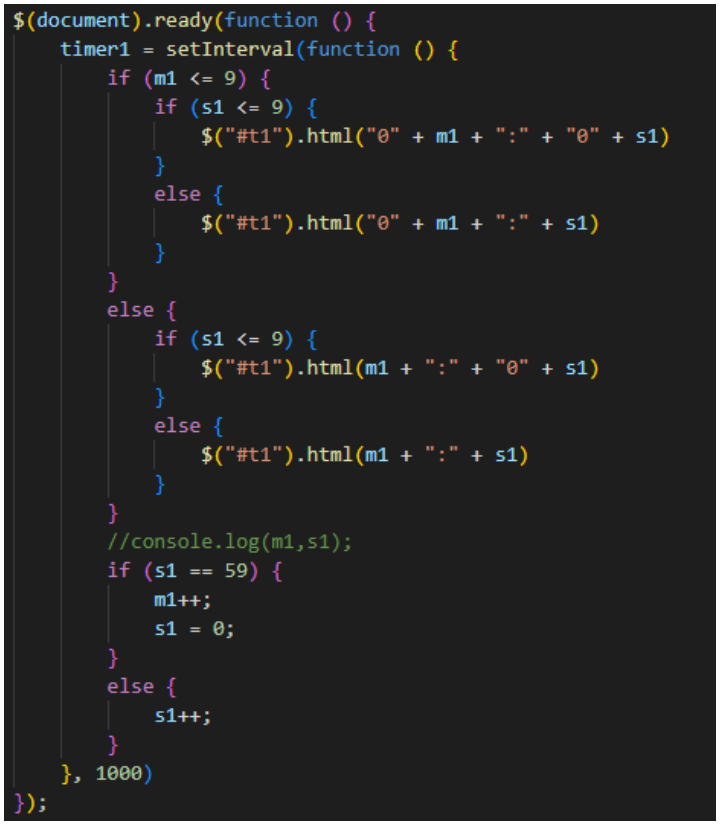
```
function odkryjpole(wierszdoodkrycia, poledoodkrycia, minydookolaodkrywanegopola) {
    $("#w" + wierszdoodkrycia + "p" + poledoodkrycia).html(minydookolaodkrywanegopola);
    switch (minydookolaodkrywanegopola) {
        case 0:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","white")
        break;
        case 1:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","blue")
        break;
        case 2:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","green")
        break;
        case 3:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","red")
        break;
        case 4:
        $("#w" + wierszdoodkrycia + "p" + poledoodkrycia).css("color", "darkblue")
        break;
        case 5:
        $("#w" + wierszdoodkrycia + "p" + poledoodkrycia).css("color", "crimson")
        break;
        case 6:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","teal")
        break;
        case 7:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","black")
        break;
        case 8:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","gray")
        break;
    }
}
```

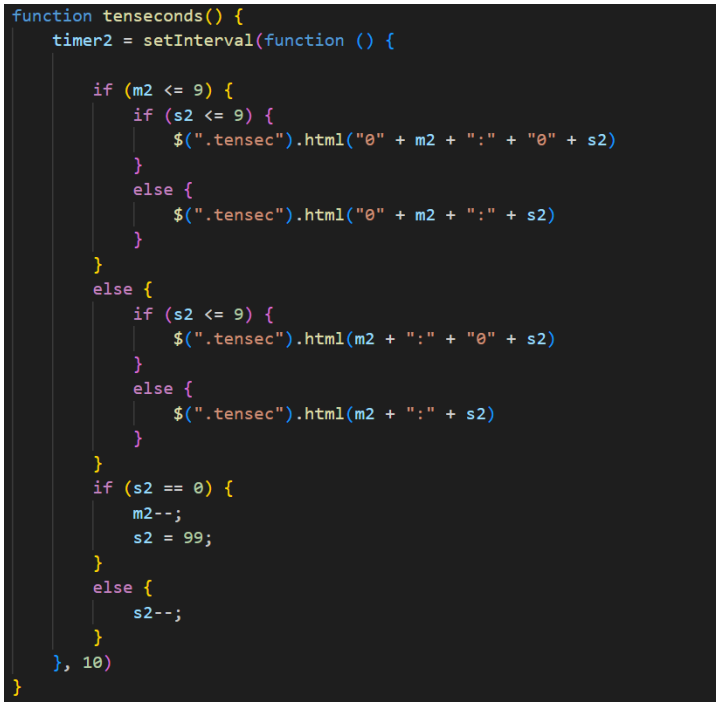
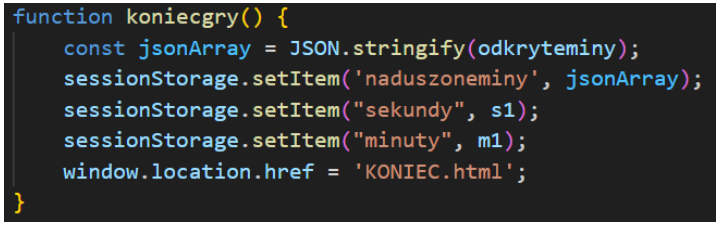
losnum	Funkcja losuje działanie do rozbrojenia bomby		 rys.17 Funkcja losnum

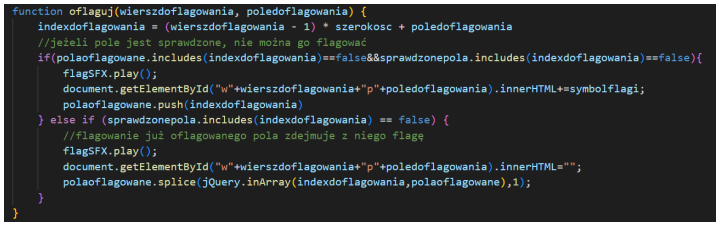
```
function losnum() {
    var a = Math.floor(Math.random() * 10) + 1;
    var b = Math.floor(Math.random() * 10) + 1;
    //var c = Math.floor(Math.random() * 0) + 4; // DLA '/'
    var c = Math.floor(Math.random() * 4) + 1;
    var isdev = false;
    console.log(c)
    var znk;
    switch (c) {
        case 1:
            znk = '+'
            odp = a + b;
            break;
        case 2:
            znk = '-'
            odp = a - b;
            break;
        case 3:
            znk = '*'
            odp = a * b;
            break;

        case 4:
            znk = '/'
            while (isdev == false) {
                if (a % b == 0) {
                    isdev = true;
                }
                else {
                    a = Math.floor(Math.random() * 10) + 1;
                }
            }
            odp = a / b;
            break;
    }

    $(".quest").html("Ile Wynosi " + a + " " + znk + " " + b + "?")


}
```

wys	Funkcja wywołuje się po naciśnięciu na pole następnie sprawdza czy pole to zawiera miny jeśli tak to gra dzwięk , wywołuje funkcję losnum i funkcję timeout . Następnie przekazuje wiersz i kolumne do tablicy naduszoneminy jeżeli inne pole z bombą zostanie naciśnięte jeżeli bomba nie została rozbrojona gra zakończy się porażką gracza. Jeżeli na polu nie ma miny wywoływana zostaje funkcja sprawdzpole	id - argument nie jest używany w- numer wiersza nacisniętego guzikia p- numer kolumny naciśniętego wiersza	 rys.18 Funkcja wys

conf	Funkcja wywoływana jest po naciśnięciu guzika sprawdzenia odpowiedzi odtwarza dźwięk w zależności od poprawności odpowiedzi jeżeli odpowiedź jest zła to gracz przenosi się na stronę główną jeżeli jest poprawna to wyłączany jest timer 10 sekund i ustawiany z powrotem na 10s następnie zmienia się kolor pola indukujący że bomba została rozbrojona jeżeli nie ma już min wywołana zostaje funkcja koniecgry		 rys.19 Funkcja conf
timeout	Funkcja ustawia 10 sekund na rozbrojenie bomby		 rys.20 Funkcja timeout

Lose	Funkcja odtwarza dźwięk eksplozji i kończy ge jeżeli gracz nie rozbroił miny na czas		 rys.21 Funkcja Lose
	Funkcja wywołuje się po włączeniu strony i ustawia timer liczący czas rozgrywki		 rys.22 Nienazwana funkcja

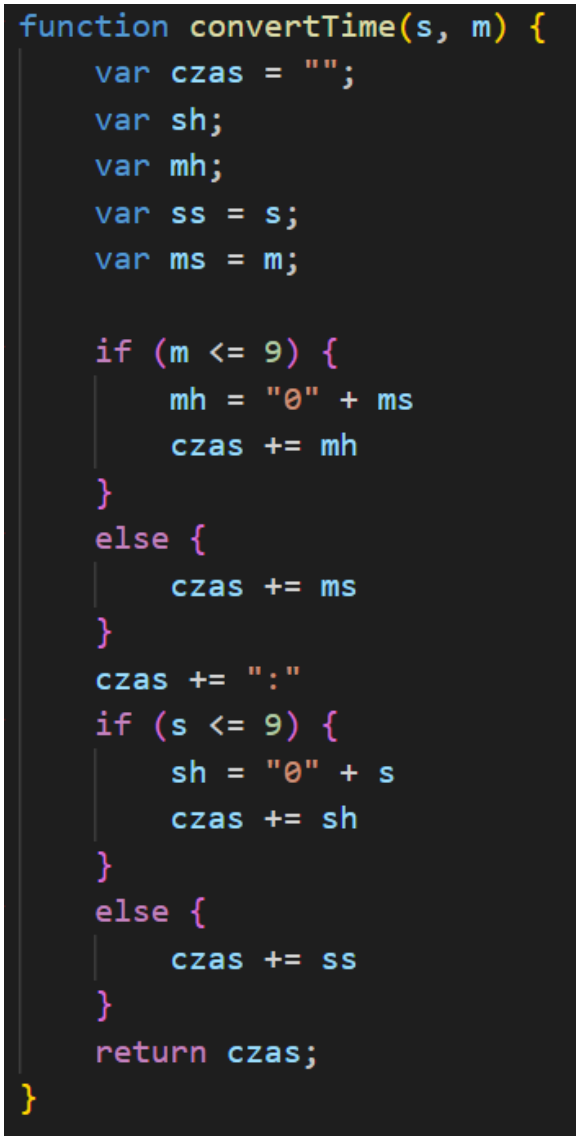
tenseconds	Funkcja odlicza od 10 sekund w dół		 rys.23 Funkcja tenseconds oraz nienazwana funkcja
koniecgry	Funkcja zbiera statystyki gracza i wysyła go na stronę z gratulacjami.		 rys.24 Funkcja koniecgry

oflaguj	Funkcja stawia lub zdejmuje znak flagi z nieodkrytych pól oraz gra odpowiednie efekty dźwiekowe	wierszdoflagowania - numer wierszu pola do olfagowania poledoflagowania - numer pola w wierszu pola do oflagowania (numer kolumny pola do oflagowania)	 rys.25 Funkcja oflaguj

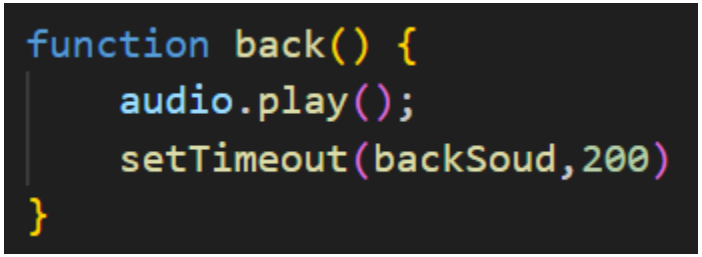
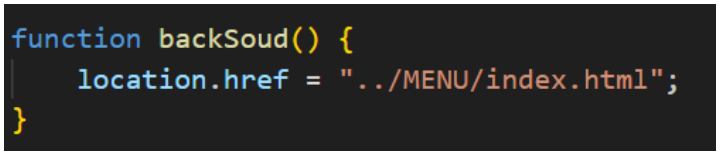
```
function oflaguj(wierszdoflagowania, poledoflagowania) {
    indexdoflagowania = (wierszdoflagowania - 1) * szerokosc + poledoflagowania
    //jeżeli pole jest sprawdzone, nie można go flagować
    if(polaoflagowane.includes(indexdoflagowania)==false&&sprawdzonepola.includes(indexdoflagowania)==false){
        flagSFX.play();
        document.getElementById("w"+wierszdoflagowania+"p"+poledoflagowania).innerHTML+=symbolflagi;
        polaoflagowane.push(indexdoflagowania)
    } else if (sprawdzonepola.includes(indexdoflagowania) == false) {
        //flagowanie już oflagowanego pola zdejmuje z niego flagę
        flagSFX.play();
        document.getElementById("w"+wierszdoflagowania+"p"+poledoflagowania).innerHTML="";
        polaoflagowane.splice(jQuery.inArray(indexdoflagowania,polaoflagowane),1);
    }
}
```

PLANSZA/endpg.js

Nazwa funkcji JavaScript	**Opis działania**	**Argumenty**	**Kod**

convertTime	Funkcja konwertuje przekazany czas aby można było go wyświetlić w formacie 00:00	s - sekundy z statystyk gracza m - minuty z statystyk gracza	 rys.26 Funkcja convertTime

```
function convertTime(s, m) {
    var czas = "";
    var sh;
    var mh;
    var ss = s;
    var ms = m;

    if (m <= 9) {
        mh = "0" + ms
        czas += mh
    }
    else {
        czas += ms
    }
    czas += ":"
    if (s <= 9) {
        sh = "0" + s
        czas += sh
    }
    else {
        czas += ss
    }
    return czas;
}
```

	Funkcja wywołuje się po włączeniu strony i wyświetla przekazane dane gry na stronie		 rys.27 Nienazwana funkcja 2
back	funkcja gra odpowiedni plik dźwiękowy następnie wywołuje funkcję backSoud.		 rys.28 Funkcja back 2
backSond	Funkcja przerzuca gracza z powrotem do menu.		 rys.29 Funkcja backSoud

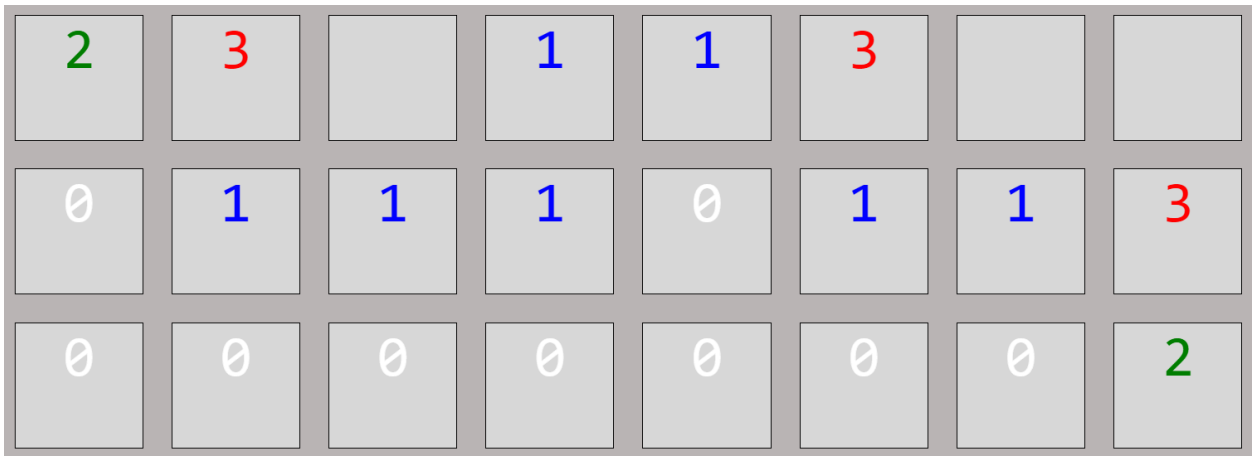
3. Instrukcja obsługi gry

Cel gry:

Celem gry jest zaznaczenie wszystkich bezpiecznych pól lub rozbrojenie wszystkich bomb. Po osiągnięciu dowolnego z tych celów gra się kończy.

Rozgrywka:

Podczas rozgrywki gracz nadusza lewym przyciskiem myszy w pola. Bezpieczne pola wyświetlają ile min znajduje się dookoła.

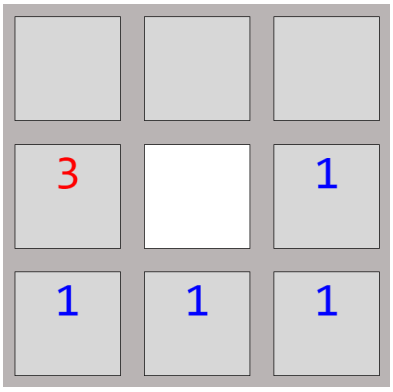


rys.30 Odkryte bezpieczne pola

Miny:

Naduszenie w minę zmusza gracza do rozbrojenia jej. Aby to zrobić gracz musi rozwiązać zadanie matematyczne w mniej niż 10 sekund. Rozbrojone miny zaznaczone są kolorem białym.

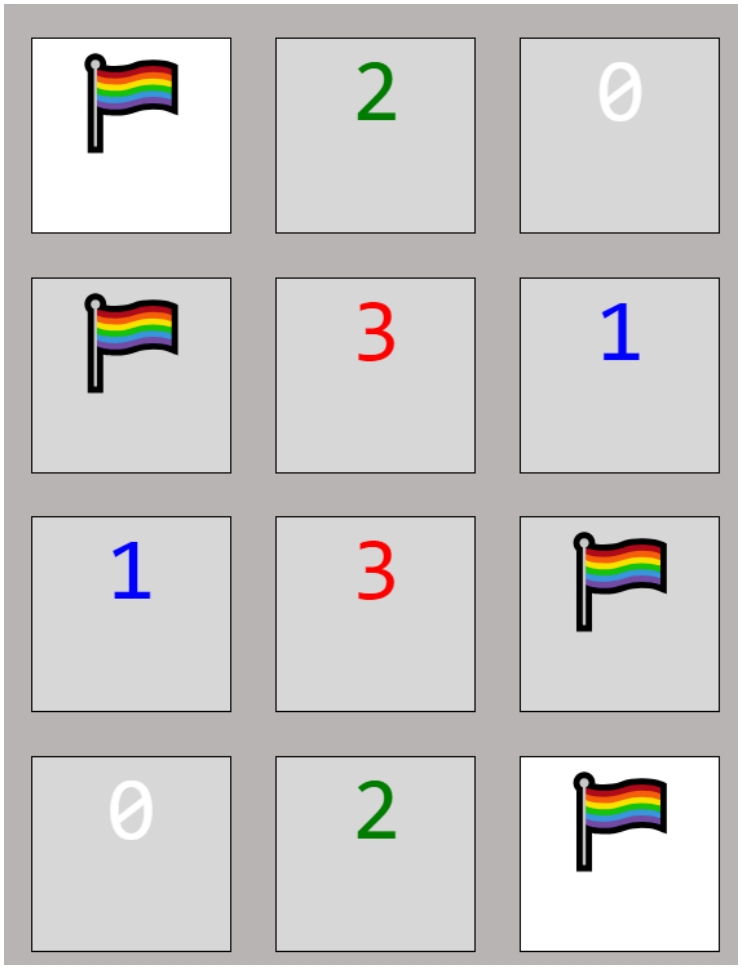
rys.31 Rozbrajacz



rys.32 Rozbrojona Mina

Flagi:

Gracz może pomagać sobie w zapamiętywaniu pozycji min przy użyciu flag. Flagi można postawić na nieodkrytych polach za pomocą prawego przycisku myszy. Aby zdjąć flagę z pola trzeba znowu kliknąć w nie prawym przyciskiem myszy



rys.33 Flagi

4. Kod źródłowy gry

MENU/scriptindex.js:

```
//definiuje SFX
var audio = new Audio("../SFX/SpaceTapped.ogg");


//wersja 'sound' każdej funkcji zapewnia że SFX zagra bez problemów
function startgame(){
    audio.play();
    setTimeout(startgameSound,200);
}
//funkcja sprawdza czy ustawienia były zmieniane. Jeżeli nie, ustawia domyślne
function startgameSound(){
    if(sessionStorage.getItem("altered")!=1){
        sessionStorage.setItem("diff", 10);
        sessionStorage.setItem("sizeh", 9);
        sessionStorage.setItem("sizew", 9);
    }
    location.href = "../PLANSZA/index.html";
}


function Ustawienia(){
    audio.play();
    setTimeout(UstawieniaSound,200);
}
//funkcja zabiera gracza na podstronę z ustawieniami
function UstawieniaSound(){
```

```
    location.href = "../MENU/settings.html"
}


function zakoncz(){
    audio.play();
    setTimeout(zakonczSound,200);
}
function zakonczSound(){
    //window.close('','_parent','');
    //^nie działa
}
```

MENU/scriptsettings.js:

```
//Definiuje SFX
var audio = new Audio("../SFX/SpaceTapped.ogg");


//sprawdzamy czy ustawienia były zmienione. Jeżeli nie, ustawiamy
podstawowe
if(sessionStorage.getItem("altered")!=1){
    var diff = 10;
    var sizew = 9;
    var sizeh = 9;
}else{
    var diff = sessionStorage.getItem("diff");
    var sizeh = sessionStorage.getItem("sizeh");
    var sizew = sessionStorage.getItem("sizew");
}
```

```
//ustawiamy odpowiednie guziki na kolor niebieski
switch(Number(diff)){
    case 10:
        //console.log(diff);
        $("#easy").css({"background-color":"rgb(58, 96, 178)"});
        break;
    case 20:
        //console.log(diff);
        $("#normal").css({"background-color":"rgb(58, 96, 178)"});
        break;
    case 30:
        //console.log(diff);
        $("#hard").css({"background-color":"rgb(58, 96, 178)"});
        break;
    case 40:
        //console.log(diff);
        $("#impossible").css({"background-color":"rgb(58, 96, 178)"});
        break;
    default:
        //console.log(diff);
        $("#wlasnydiff").css({"background-color":"rgb(58, 96, 178)"});
}

switch(sizeh+"|"+sizew){
    case ("9|9"):
        //console.log(sizeh);
```

```
        //console.log(sizew);
        $("#9x9").css({"background-color":"rgb(58, 96, 178)"});
        break;
    case ("12|12"):
        //console.log(sizeh);
        //console.log(sizew);
        $("#12x12").css({"background-color":"rgb(58, 96, 178)"});
        break;
    case ("15|15"):
        //console.log(sizeh);
        //console.log(sizew);
        $("#15x15").css({"background-color":"rgb(58, 96, 178)"});
        break;
    case ("18|18"):
        //console.log(sizeh);
        //console.log(sizew);
        $("#18x18").css({"background-color":"rgb(58, 96, 178)"});
        break;
    default:
        //console.log(sizeh);
        //console.log(sizew);
        $("#wlasnysize").css({"background-color":"rgb(58, 96, 178)"});
}


//funkcja zmienia kolor guzików i ustawia nową trudność
function fdiff(x){
    diff=x;
```

```
    audio.play();
    $(".btndiff").css({"background-color":"rgb(98, 136, 218)"});
    switch(diff){
        case 10:
            $("#easy").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case 20:
            $("#normal").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case 30:
            $("#hard").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case 40:
            $("#impossible").css({"background-color":"rgb(58, 96, 178)"});
            break;
    }
};


//funkcja zmienia kolor guzików i ustawia nowy rozmiar
function fsize(x){
    sizew=x;
    sizeh=x;
    audio.play();
    $(".btnsize").css({"background-color":"rgb(98, 136, 218)"});
    switch(sizeh+"|"+sizew){
        case ("9|9"):
            $("#9x9").css({"background-color":"rgb(58, 96, 178)"});
```

```
            break;
        case ("12|12"):
            $("#12x12").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case ("15|15"):
            $("#15x15").css({"background-color":"rgb(58, 96, 178)"});
            break;
        case ("18|18"):
            $("#18x18").css({"background-color":"rgb(58, 96, 178)"});
            break;
    }
};

//funkcja odpowiedzialna za ustawianie własnego rozmiaru planszy
function customsize(){
    audio.play();
    sizeh = prompt("Podaj wysokość planszy [2-200]");
    //pytanie jest powtarzane dopóki odpowiedź gracza nie znajdzie się w
podanym zakresie
    while(Number(sizeh)>200||Number(sizeh)<2){
        sizeh = prompt("Podaj wysokość planszy [2-200]");
    }
    sizew = prompt("Podaj szerokość planszy [2-200]");
    while(Number(sizew)>200||Number(sizew)<2){
        sizew = prompt("Podaj szerokość planszy [2-200]");
    }
    $(".btnsize").css({"background-color":"rgb(98, 136, 218)"});
```

```
    $("#wlasnysize").css({"background-color":"rgb(58, 96, 178)"});
}


//funkcja odpowiedzialna za ustawianie własnego poziomu trudności
function customdiff(){
    audio.play();
    diff = prompt("Podaj procent zaminowanych pól [1-99]");
    while(Number(diff)>99||Number(diff)<1){
        diff = prompt("Podaj procent zaminowanych pól [1-99]");
    }
    $(".btndiff").css({"background-color":"rgb(98, 136, 218)"});
    $("#wlasnydiff").css({"background-color":"rgb(58, 96, 178)"});
}


//funkcja zapisuje nowe ustawienia i fakt że były one zmienione
function save(){
    audio.play();
    sessionStorage.setItem("diff", diff);
    sessionStorage.setItem("sizew", sizew);
    sessionStorage.setItem("sizeh", sizeh);
    sessionStorage.setItem("altered", 1);
};


//funkcja wysyła gracza z powrotem na stronę główną. wersja 'sound'
zapewnia że SFX zagra bez przeszkód
function back(){
    audio.play();
```

```
    setTimeout(backSound,200)
}
function backSound(){
    window.location.href = '../MENU/index.html';
}
```

PLANSZA/script.js

```
//Definiuje SFX
var poleSFX = new Audio("../SFX/SpaceTapped.ogg");
var minaSFX = new Audio("../SFX/MineTapped.ogg");
var odpSFX = new Audio("../SFX/AnwserSubmitted.ogg");
var looseSFX = new Audio("../SFX/YouLoose.ogg");
var flagSFX = new Audio("../SFX/Flag.ogg");

//dane sa pobierane z session storage-u
var wysokosc = Number(sessionStorage.getItem("sizeh"));
var szerokosc = Number(sessionStorage.getItem("sizew"));
var mnoznik = Number(sessionStorage.getItem("diff") / 100);
var iloscmin = Number(Math.round((wysokosc * szerokosc) * mnoznik));
//wypisuje dane obok planszy
//document.getElementById("datawys").innerHTML=wysokosc;
//document.getElementById("dataminy").innerHTML=iloscmin;
//document.getElementById("dataszer").innerHTML=szerokosc;
$("#t2").html(iloscmin)
$("#t3").html(wysokosc * szerokosc - iloscmin)
```

```
//definiuje zmienne potrzebne dla utworzenia planszy

const plansza = document.getElementById("plansza");

var sprawdzonepola = [];

var topaste = " ";


//pętla tworząca plansze na podstawie danych z session starage-u

for (i = 1; i <= wysokosc; i++) {

    topaste += "<div class='wiersz' id='w" + i + "'>";

    for (j = 1; j <= szerokosc; j++) {

        topaste += "<div class='pole' id='w" + i + "p" + j + "' onclick =
'wys(w" + i + "p" + j + "," + i + "," + j + ")' oncontextmenu='oflaguj(" +
i + "," + j + ")'></div>";

    }

    topaste += "</div>"

}


//wypisanie planszy

plansza.innerHTML += topaste;


//wyliczam rozmiar pól na planszy

polewymiarywys = Math.round(100 / wysokosc);

polewymiaryszer = Math.round(100 / szerokosc);


//wyświetlam pola na planszy

$(".wiersz").css({ "height": polewymiarywys - 2 + "%" });

$(".pole").css({ "background-color": "rgb(216, 216, 216)", "border": "1px
solid black", "margin": "1%", "height": "98%", "width": polewymiaryszer -
2 + "%", "text-align": "center", "font-size": (polewymiaryszer * 32) +
"%", "font-family": "monospace", "vertical-align": "baseline" });
```

```
//definiuje zmienne potrzebne do losowania pozycji min
var miny = [];
var miny2 = [];
var randminapole;
var randminawiersz;
var minydowylosowania = iloscmin;

//losuje pozycje min
while (minydowylosowania > 0) {
    randminawiersz = Math.floor(Math.random() * wysokosc + 1);
    randminapole = Math.floor(Math.random() * szerokosc + 1);
    if (miny.includes(Number((randminawiersz - 1) * szerokosc +
randminapole)) == false) {
        miny2.push({ "wiersz": randminawiersz, "pole": randminapole,
"index": ((randminawiersz - 1) * szerokosc + randminapole) });
        miny.push((randminawiersz - 1) * szerokosc + randminapole);
        minydowylosowania--;
    }
}

//wyświetlam miny na planszy (używane podczas testów)
/*
for(i=0;i<iloscmin;i++){

$("#w"+miny2[i].wiersz+"p"+miny2[i].pole).css({"background-color":"white"}
);
}
```

```
*/



//sprwadzpole jest wywolywane kiedy juz wiemy ze pole nie jest bombą
function sprawdzpole(wierszwybranegopola, polewybranegopola,
indexwybranegopola) {


    //console.log("sprawdzam pole o indexie "+indexwybranegopola);
    var minydookolawybranegopola = 0;
    //jeżeli pole nie było już sprawdzane, liczymy ile pól dookoła są
minami
    if(sprawdzonepola.includes(Number(indexwybranegopola))==false){
        poleSFX.play();

if(miny.includes(Number(indexwybranegopola-szerokosc-1))!=false&&wierszwyb
ranegopola-1>0&&polewybranegopola-1>0){
                minydookolawybranegopola++;
                //console.log("mina w checku 1:
"+Number(indexwybranegopola-szerokosc-1));
            }

if(miny.includes(Number(indexwybranegopola-szerokosc))!=false&&wierszwybra
negopola-1>0){
                minydookolawybranegopola++;
                //console.log("mina w checku 2:
"+Number(indexwybranegopola-szerokosc));
            }

if(miny.includes(Number(indexwybranegopola-szerokosc+1))!=false&&wierszwyb
ranegopola-1>0&&polewybranegopola+1<=szerokosc){
```

```
                minydookolawybranegopola++;
                //console.log("mina w checku 3:
"+Number(indexwybranegopola-szerokosc+1));
            }

if(miny.includes(Number(indexwybranegopola-1))!=false&&polewybranegopola-1
>0){
                minydookolawybranegopola++;
                //console.log("mina w checku 4:
"+Number(indexwybranegopola-1));
            }

if(miny.includes(Number(indexwybranegopola+1))!=false&&polewybranegopola+1
<=szerokosc){
                minydookolawybranegopola++;
                //console.log("mina w checku 5:
"+Number(indexwybranegopola+1));
            }

if(miny.includes(Number(indexwybranegopola+szerokosc-1))!=false&&wierszwyb
ranegopola+1<=wysokosc&&polewybranegopola-1>0){
                minydookolawybranegopola++;
                //console.log("mina w checku 6:
"+Number(indexwybranegopola+szerokosc-1));
            }

if(miny.includes(Number(indexwybranegopola+szerokosc))!=false&&wierszwybra
negopola+1<=wysokosc){
                minydookolawybranegopola++;
                //console.log("mina w checku 7:
"+Number(indexwybranegopola+szerokosc));
```

```
            }

if(miny.includes(Number(indexwybranegopola+szerokosc+1))!=false&&wierszwyb
ranegopola+1<=wysokosc&&polewybranegopola+1<=szerokosc){
                minydookolawybranegopola++;
                //console.log("mina w checku
8"+Number(indexwybranegopola+szerokosc+1));
            }
        //po zliczeniu min, dodaję numer pola do listy sprawdzonych pól,
odświeżam licznik i wyświetlam numer min na polu
        sprawdzonepola.push(Number((wierszwybranegopola - 1) * szerokosc +
polewybranegopola));
        $("#t3").html(wysokosc * szerokosc - iloscmin -
sprawdzonepola.length)
        //console.log("sprawdzono pole o indexie "+indexwybranegopola+"
znaleziono "+minydookolawybranegopola+" min");
        odkryjpole(wierszwybranegopola, polewybranegopola,
minydookolawybranegopola);
        //jeżeli dookoła nie ma min, pola dookoła też są sprawdzane
        if (minydookolawybranegopola == 0) {
            sprawdzpoladookola(wierszwybranegopola, polewybranegopola,
indexwybranegopola);
        }
        //jeżeli nie ma nieodkrytych pól które nie są minami, gra się
kończy
        if (wysokosc * szerokosc - iloscmin - sprawdzonepola.length == 0)
{
            koniecgry();
        }
    } else {
```

```
        //console.log("wiersz:"+wierszwybranegopola+"
pole:"+polewybranegopola+" już było sprawdzone")

    }

}



//funkcja sprawdza pola dookoła wybranego pola

function sprawdzpoladookola(wierszoryginalnegopola, poleoryginalnegopola,
indexoryginalnegopola) {

    //jeżeli pole jest miną, poza planszą lub było już sprawdzane, nie
będzie ono sprawdzane

    if (miny.includes(indexoryginalnegopola - szerokosc - 1) == false &&
wierszoryginalnegopola - 1 > 0 && poleoryginalnegopola - 1 > 0 &&
sprawdzonepola.includes(indexoryginalnegopola - szerokosc - 1) == false &&
indexoryginalnegopola - szerokosc - 1 > 0) {

        sprawdzpole(wierszoryginalnegopola - 1, poleoryginalnegopola - 1,
Number(indexoryginalnegopola - szerokosc - 1));

    }

    if (miny.includes(indexoryginalnegopola - szerokosc) == false &&
wierszoryginalnegopola - 1 > 0 &&
sprawdzonepola.includes(indexoryginalnegopola - szerokosc) == false &&
indexoryginalnegopola - szerokosc > 0) {

        sprawdzpole(wierszoryginalnegopola - 1, poleoryginalnegopola,
Number(indexoryginalnegopola - szerokosc));

    }

    if (miny.includes(indexoryginalnegopola - szerokosc + 1) == false &&
wierszoryginalnegopola - 1 > 0 && poleoryginalnegopola + 1 <= szerokosc &&
sprawdzonepola.includes(indexoryginalnegopola - szerokosc + 1) == false &&
indexoryginalnegopola - szerokosc + 1 > 0) {

        sprawdzpole(wierszoryginalnegopola - 1, poleoryginalnegopola + 1,
Number(indexoryginalnegopola - szerokosc + 1));

    }
```

```
    if (miny.includes(indexoryginalnegopola - 1) == false &&
poleoryginalnegopola - 1 > 0 &&
sprawdzonepola.includes(indexoryginalnegopola - 1) == false &&
indexoryginalnegopola - 1 > 0) {

        sprawdzpole(wierszoryginalnegopola, poleoryginalnegopola - 1,
Number(indexoryginalnegopola - 1));

    }

    if (miny.includes(indexoryginalnegopola + 1) == false &&
poleoryginalnegopola + 1 <= szerokosc &&
sprawdzonepola.includes(indexoryginalnegopola + 1) == false &&
indexoryginalnegopola + 1 > 0) {

        sprawdzpole(wierszoryginalnegopola, poleoryginalnegopola + 1,
Number(indexoryginalnegopola + 1));

    }

    if (miny.includes(indexoryginalnegopola + szerokosc - 1) == false &&
wierszoryginalnegopola + 1 <= wysokosc && poleoryginalnegopola - 1 > 0 &&
sprawdzonepola.includes(indexoryginalnegopola + szerokosc - 1) == false &&
indexoryginalnegopola + szerokosc - 1 > 0) {

        sprawdzpole(wierszoryginalnegopola + 1, poleoryginalnegopola - 1,
Number(indexoryginalnegopola + szerokosc - 1));

    }

    if (miny.includes(indexoryginalnegopola + szerokosc) == false &&
wierszoryginalnegopola + 1 <= wysokosc &&
sprawdzonepola.includes(indexoryginalnegopola + szerokosc) == false &&
indexoryginalnegopola + szerokosc > 0) {

        sprawdzpole(wierszoryginalnegopola + 1, poleoryginalnegopola,
Number(indexoryginalnegopola + szerokosc));

    }

    if (miny.includes(indexoryginalnegopola + szerokosc + 1) == false &&
wierszoryginalnegopola + 1 <= wysokosc && poleoryginalnegopola + 1 <=
szerokosc && sprawdzonepola.includes(indexoryginalnegopola + szerokosc +
1) == false && indexoryginalnegopola + szerokosc + 1 > 0) {
```

```
        sprawdzpole(wierszoryginalnegopola + 1, poleoryginalnegopola + 1,
Number(indexoryginalnegopola + szerokosc + 1));

    }

}


//funkcja wypisuje ilość min dookoła na polu

function odkryjpole(wierszdoodkrycia, poledoodkrycia,
minydookolaodkrywanegopola) {

    $("#w" + wierszdoodkrycia + "p" +
poledoodkrycia).html(minydookolaodkrywanegopola);

    switch (minydookolaodkrywanegopola) {

        case 0:

        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","white")

        break;

        case 1:

        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","blue")

        break;

        case 2:

        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","green")

        break;

        case 3:

        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","red")

        break;

        case 4:

        $("#w" + wierszdoodkrycia + "p" + poledoodkrycia).css("color",
"darkblue")

        break;

        case 5:
```

```
        $("#w" + wierszdoodkrycia + "p" + poledoodkrycia).css("color",
"crimson")
        break;
        case 6:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","teal")
        break;
        case 7:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","black")
        break;
        case 8:
        $("#w"+wierszdoodkrycia+"p"+poledoodkrycia).css("color","gray")
        break;
    }
}


// losowanie numerow , znakow i odpowiedzi
function losnum() {
    var a = Math.floor(Math.random() * 10) + 1;
    var b = Math.floor(Math.random() * 10) + 1;
    //var c = Math.floor(Math.random() * 0) + 4; // DLA '/'
    var c = Math.floor(Math.random() * 4) + 1;
    var isdev = false;
    console.log(c)
    var znk;
    switch (c) {
        case 1:
            znk = '+'
```

```
            odp = a + b;
            break;
        case 2:
            znk = '-'
            odp = a - b;
            break;
        case 3:
            znk = '*'
            odp = a * b;
            break;

        case 4:
            znk = '/'
            while (isdev == false) {
                if (a % b == 0) {
                    isdev = true;
                }
                else {
                    a = Math.floor(Math.random() * 10) + 1;
                }
            }
            odp = a / b;
            break;
    }

    $(".quest").html("Ile Wynosi " + a + " " + znk + " " + b + "?")
```

```
}
var naduszoneminy = [];
var time;
var odkryteminy = [];
// on click na polach
function wys(id, w, p) {
    // trafiono na mine
    //console.log(w,p)
    if (miny.includes((w - 1) * szerokosc + p) == true &&
odkryteminy.includes("w" + w + "p" + p) == false) {
        //console.log("hit")
        minaSFX.play();
        losnum();
        timeout();
        naduszoneminy.push("w" + w + "p" + p);
        if (naduszoneminy.length > 1) {
            if (confirm("Przegrywasz")) {
                window.location.href = '../MENU/index.html';
            } else {
                window.location.href = '../MENU/index.html';
            }
        }
    } else if (odkryteminy.includes("w" + w + "p" + p) == false) {
sprawdzpole(w, p, (w - 1) * szerokosc + p) };
}
// Funkcja sprawdzająca prawidziwosc odpowiedzi
function conf() {
```

```
    var value = $("#odp").val();
    // dobra odp
    if (value == odp) {
        //console.log("fin")
        odpSFX.play();
        $("#odp").val(" ")
        $(".quest").html("...")
        clearTimeout(time);
        clearInterval(timer2);
        s2 = 0;
        m2 = 10;
        $(".tensec").html("10:00")
        $("#" + naduszoneminy[0]).css({ "background-color": "white" });
        odkryteminy.push(naduszoneminy[0]);
        $("#t2").html(iloscmin - odkryteminy.length);
        naduszoneminy.shift();
        if (iloscmin - odkryteminy.length == 0) {
            koniecgry();
        }
    }
    //zla odp
    else if(value!=odp){
        looseSFX.play();
        if(confirm("Przegrywasz")){
            window.location.href = '../MENU/index.html';
        }else{
            window.location.href = '../MENU/index.html';
```

```
        }
    }
}
// ustawianie czasu na rozbrojenie
function timeout() {
    time = setTimeout(Lose, 10020)
    tenseconds();
    // console.log("time on")

}
// funkcja wywolywana gdy konczy sie czas na rozbrojenie
function Lose(){
    looseSFX.play();
    console.log("time off");
    if (confirm("Rozbrojenie bomby zajeło zbyt długo czasu. Przegrywasz"))
{
        window.location.href = '../MENU/index.html';
    } else {
        window.location.href = '../MENU/index.html';
    }
}
var timer1;
// zmienne sekund i minut
var s1 = 1;
var m1 = 0;
// timer gry
$(document).ready(function () {
```

```
    timer1 = setInterval(function () {
        if (m1 <= 9) {
            if (s1 <= 9) {
                $("#t1").html("0" + m1 + ":" + "0" + s1)
            }
            else {
                $("#t1").html("0" + m1 + ":" + s1)
            }
        }
        else {
            if (s1 <= 9) {
                $("#t1").html(m1 + ":" + "0" + s1)
            }
            else {
                $("#t1").html(m1 + ":" + s1)
            }
        }
        //console.log(m1,s1);
        if (s1 == 59) {
            m1++;
            s1 = 0;
        }
        else {
            s1++;
        }
    }, 1000)
});
```

```
// funkcja zajmująca się odliczaniem do wybuchu bomby (brak implementacji)
var s2 = 0;
var m2 = 10;
var timer2;
function tenseconds() {
    timer2 = setInterval(function () {

        if (m2 <= 9) {
            if (s2 <= 9) {
                $(".tensec").html("0" + m2 + ":" + "0" + s2)
            }
            else {
                $(".tensec").html("0" + m2 + ":" + s2)
            }
        }
        else {
            if (s2 <= 9) {
                $(".tensec").html(m2 + ":" + "0" + s2)
            }
            else {
                $(".tensec").html(m2 + ":" + s2)
            }
        }
        if (s2 == 0) {
            m2--;
            s2 = 99;
        }
```

```
        else {
            s2--;
        }
    }, 10)
}
//funkcja kończąca grę kiedy nie ma żadnych nieodkrytych min lub
bezpiecznych pól, wysyła gracza na stronę z gratulacjami
function koniecgry() {
    const jsonArray = JSON.stringify(odkryteminy);
    sessionStorage.setItem('naduszoneminy', jsonArray);
    sessionStorage.setItem("sekundy", s1);
    sessionStorage.setItem("minuty", m1);
    window.location.href = 'KONIEC.html';
}

//symbol flagi do zaznaczania min zmienia się w zależności od miesiąca
var symbolflagi;
const datetime = new Date();
switch (datetime.getMonth() + 1) {
    case 1:
        //w styczniu jest to kawałek ciasta ponieważ Łukasz ma w styczniu
urodziny
        symbolflagi = "🍰";
        break;
    case 2:
        //w lutym jest to serce z okazji walentynek
        symbolflagi = "❤";
        break;
```

```
    case 3:
        //w marcu jest to ciasto ponieważ Paweł ma w styczniu urodziny. Warto zauważyć że Paweł dostaje całe ciasto a Łukasz tylko kawałek. To dlatego, że urodziny Pawła są ważniejsze.
        symbolflagi = "🎂";
        break;
    case 4:
        //w kwietniu jest to klaun z okazji prima aprilis
        symbolflagi = "🤡";
        break;
    case 5:
        //w maju jest to polska flaga z okazji rocznicy podpisania konstytucji
        symbolflagi = "🇵🇱";
        break;
    case 6:
        //w czerwcu jest to tęczowa flaga z okazji miesiąca równości
        symbolflagi = "🏳️‍🌈";
        break;
    case 7:
        //w lipcu jest to słońce z uwagi na słoneczną lipcową pogodę
        symbolflagi = "☀️";
        break;
    case 8:
        //w sierpniu jest to strzałka w lewo z okazji międzynarodowego dnia osób leworęcznych
        symbolflagi = "⬅️";
        break;
```

```
    case 9:
        //we wrześniu jest to szkoła z okazji rozpoczęcia roku szkolnego
        symbolflagi = "🏫";
        break;
    case 10:
        //we październiku jest to dynia z okazji halloween
        symbolflagi = "🎃";
        break;
    case 11:
        //we listopadzie jest to parasol z uwagi na jesienną pogodę
        symbolflagi = "☔";
        break;
    case 12:
        //w grudniu jest to choinka z okazji świąt Bożego narodzenia
        symbolflagi = "🎄";
        break;
    default:
        //domyślnie jest to czerwona flaga
        symbolflagi = "🚩";
}


//kod który sprawia że prawy przycisk nie otwiera menu
document.addEventListener('contextmenu', event => {
    event.preventDefault();
});


//funkcja i zmienne odpowiedzialne za flagowanie pól
```

```
var polaoflagowane = [];

var indexdoflagowania;

function oflaguj(wierszdoflagowania, poledoflagowania) {

    indexdoflagowania = (wierszdoflagowania - 1) * szerokosc +
poledoflagowania

    //jeżeli pole jest sprawdzone, nie można go flagować


if(polaoflagowane.includes(indexdoflagowania)==false&&sprawdzonepola.inclu
des(indexdoflagowania)==false){

        flagSFX.play();


document.getElementById("w"+wierszdoflagowania+"p"+poledoflagowania).inner
HTML+=symbolflagi;

        polaoflagowane.push(indexdoflagowania)

    } else if (sprawdzonepola.includes(indexdoflagowania) == false) {

        //flagowanie już oflagowanego pola zdejmuje z niego flagę

        flagSFX.play();


document.getElementById("w"+wierszdoflagowania+"p"+poledoflagowania).inner
HTML="";


polaoflagowane.splice(jQuery.inArray(indexdoflagowania,polaoflagowane),1);

    }

}
```

PLANSZA/endpg.js

```
//definiuje SFX

var audio = new Audio("../SFX/SpaceTapped.ogg");


var wysokosc = Number(sessionStorage.getItem("sizeh"));
```

```
var szerokosc = Number(sessionStorage.getItem("sizew"));
var diff = Number(sessionStorage.getItem("diff"));
var difstr;
var minystr = sessionStorage.getItem('naduszoneminy');
var miny = JSON.parse(minystr);
var sekundy = Number(sessionStorage.getItem("sekundy"));
var minuty = Number(sessionStorage.getItem("minuty"));
//console.log(sekundy)
//console.log(minuty);
// zamiana liczbowego poziomu trudonosci na napis
switch (diff) {
    case 10:
        diffstr = "Łatwym";
        break;
    case 20:
        diffstr = "Normalnym";
        break;
    case 30:
        diffstr = "Trudnym";
        break;
    case 40:
        diffstr = "Niemożliwym";
        break;
    default:
        diffstr = "Własnym";
        break;
```

```
}
// funkcja konwertujuaca czas aby mozna bylo go wyswietlic
function convertTime(s, m) {
    var czas = "";
    var sh;
    var mh;
    var ss = s;
    var ms = m;

    if (m <= 9) {
        mh = "0" + ms
        czas += mh
    }
    else {
        czas += ms
    }
    czas += ":"
    if (s <= 9) {
        sh = "0" + s
        czas += sh
    }
    else {
        czas += ss
    }
    return czas;
}
```

```
$(document).ready(function () {


    $(".lilh").html("Udało ci się ukończyć grę
w POLE MINOWE na poziomie " + "<d><b>" + diffstr +
"</b></d>");

    $("#miny").html(miny.length);

    $("#czas").html(convertTime(sekundy, minuty));

    $("#rozmiar").html(wysokosc + "x" + szerokosc);


});


function back() {

    audio.play();

    setTimeout(backSoud,200)

}


function backSoud() {

    location.href = "../MENU/index.html";

}
```

5. Wnioski końcowe

Część wspólna:

Z przykrością musimy stwierdzić że nie udało nam się osiągnąć wszystkich założeń projektu. Pomimo skomponowania ośmiu piosenek dla ścieżki dźwiękowej gry nie bylibyśmy w stanie faktycznie odtwarzać ich w tle bez dostania się wpierw na białą listę stron z pozwoleniem na automatyczne uruchamianie plików dźwiękowych bez wcześniejszego inputu od użytkownika. Poza tym, efekt końcowy projektu jest zgodny z wymaganiami i naszą wizją. Jesteśmy dumni z naszej współpracy.

Paweł:

Wszystkie problemy które napotkałem podczas pracy byłem w stanie rozwiązać całkiem szybko ale gdybym musiał nazwać jeden to byłaby to optymalizacja algorytmu do odkrywania pól. We wczesnych fazach projektu często był on przeładowywany i przeglądarki przerywały funkcję. Rozwiązałem to poprzez dodanie wielu nowych warunków i wymagań dzięki którym każda funkcja była wywoływana o wiele mniej razy.

Gdybym miał coś zmienić w aplikacji dostosowałbym skalowanie UI oraz zmniejszył maksymalny rozmiar planszy. Okazuje się że algorytm, pomimo wszystkich usprawnień, nadal może być przeładowany przy 40 000 polach i 40 minach.

Łukasz:

Realizacja projektu nie sprawiła mi większych trudności , jeżeli pojawiały się jakieś problemy były szybko rozwiązywane . Projekt nauczył mnie że warto zwracać uwagę na jednolitość stosowanych rozwiązań ponieważ spowalnia to pracę nad projektem gdy zaczyna się pracę po przerwie. Jeżeli miałbym czas ulepszyć nasz produkt stworzyłbym możliwość większej personalizacji strony i rozgrywki. Pomimo tego jestem zadowolony z końcowego wyglądu projektu.